MASTER EN SCIENCES INFORMATIQUES

Simulation quantitative en robotique

Conrard, E.; Darte, D.

Award date:
1989

Awarding institution:
Universite de Namur

Simulation quantitative en robotique

Partie 1

E. CONRARD D. DARTE

Mémoire présenté

en vue de l'obtention du titre

Licencié et Maître en sciences

option informatique

Septembre 1989

Institut d'informatique des Facultés Notre-Dame de la Paix de Namur

Résumé

Le sujet de ce mémoire est la réalisation d'un simulateur de robot basé sur un modèle mathématique. Ce modèle est composé de transformations graphiques implémentées par des matrices. La spécificité de ce logiciel relève de la volonté de représenter un environnement de travail le plus réaliste possible tout en prenant en considération que certaines limites sont à respecter(temps d'affichage du robot à l'écran, ...). L'interprétation des résultats est facilitée par l'emploi d'une interface graphique (les mouvements d'un bras d'un robot peuvent être vus à l'écran). ce qui rend son utilisation accessible aussi bien par des amateurs que par des experts.

Abstract

The subject of this dissertation is the realization of a robot's simulator based upon a mathematical model. This model is composed of graphics transformations, and are implemented by matrices. The software's particularity consists of representing a most realistic work environment but in the same time, it takes into account that some limits have to be respected (time to display objects on screen,...). The interpretation of the results issued by the simulation is made easier by using a graphic interface, so that it can be used by engineers, experts and novices.

Remerciements

Au terme de ce mémoire, nos plus vifs remerciements sont tout d'abord adressés à Monsieur S. Antunes, professeur à l'Escola Nautica Infante D. Henrique à Lisbonne.

Grâce à lui, nous avons eu l'occasion de passer une période de stage enrichissante tant du point de vue professionnel qu'humain. En outre, il nous a aidés dans le développement d'un logiciel qui nous a permis d'approfondir les connaissances acquises durant nos études.

Nous remercions tout autant Monsieur J. Barreto, grâce à qui notre stage ainsi que notre mémoire ont pu exister. De plus, il nous fut d'un précieux secours tant lors de la conception du logiciel que lors de la rédaction de ce mémoire.

Notre reconnaissance va également à tous ceux qui, de loin ou de près, nous ont aidés à mener notre tâche à son terme.

Enfin, nous voulons remercier tous les membres du corps académique de l'Institut d'Informatique des Facultés Notre-Dame de la Paix de Namur pour l'encadrement exceptionnel qu'ils fournissent aux étudiants.

Table des matières

PARTIE 1

0. Introduction

1. La simulation quantitative
   - 1.1 Définition de la simulation
   - 1.2 Définition d'un modèle
   - 1.3 Buts d'une étude de simulation
   - 1.4 Modélisation qualitative et quantitative
   - 1.5 Simulateur de robot et simulation quantitative
   - 1.6 Simulation et graphisme

2. Présentation de la robotique
   - 2.1 Introduction
   - 2.2 Qu'est-ce qu'un robot ?
   - 2.3 Robot : définition
   - 2.4 Les composants d'un robot
     - 2.4.1 Bras du robot
     - 2.4.2 L'organe actif
   - 2.5 Programmation d'un robot
     - 2.5.1 Introduction
     - 2.5.2 Travail réalisé par un robot : exemple
     - 2.5.3 Développement du programme d'un robot
       - 2.5.3.1 Définition du travail
       - 2.5.3.2 Conception de l'organe actif
       - 2.5.3.3 Développemnt proprement du programme
         - (A) Programmation on-line
           - (A.1) Programmation par démonstration
           - (A.2) Programmation par boîtier d'apprentissage
         - (B) Programmation off-line
     - 2.5.4 Documentation
   - 2.6 Différents modes de programmation d'un robot
   - 2.7 Teach pendant du robot RHINO-XR2
     - 2.7.1 Le teach pendant
     - 2.7.2 Fonctions offertes
   - 2.8 Teach pendant du MICROBOT teach mover
   - 2.9 Teach pendant du simulateur du robot

3 Concepts
3.1 Système d'axes
3.1.1 Système d'axes du référentiel écran
3.1.2 Système d'axes du référentiel actif
3.1.2.1 Système d'axes du robot
3.1.2.2 Système d'axes des bras suivants
3.1.2.3 Définition de la position du bras dans son système d'axes
3.1.3 Changement du point de vue
3.2 Définition du robot modélisé
3.2.1 Représentation du robot
3.2.2 Définition du bras
3.2.3 L'organe actif
3.2.4 Point fixe et surface d'appui
3.2.5 Rotule et charnière
3.2.5.1 Les rotules
3.2.5.2 Les charnières
3.2.6 Dessin d'un bras
3.3 Calcul des coordonnées
3.3.1 Calcul des coordonnées des points positionnant le bras dans son référentiel actif
3.3.2 Calcul des coordonnées des points positionnant le bras dans le référentiel écran
3.3.2.1 Calcul d'un référentiel du robot
3.3.2.2 Passage du référentiel du robot au référentiel de l'écran
3.4 Description des mouvements
3.4.1 Rotation d'un bras
3.4.2 Pivotage d'un bras
3.4.3 Translation du robot
3.5 Positions particulères
3.5.1 Position initiale
3.5.2 Position intermédiaire
4. Programme résident
4.1 Un operating system multi-tâche
4.2 Concepts de programme résident
4.3 Que se passe - t-il après un KEEP ?
4.4 Permutation des registres
4.5 Garder traces

4.6 Activation et communication avec le programme résident : explication

4.7 Connaissance du numéro d'interruption par le programme de l'utilsateur

4.8 Erreurs critiques au niveau du Dos
- 4.8.1 Erreurs critiques
- 4.8.2 Contrôl break

4.9 Déchargement du programme résident

5 Interface programme de commandes - interpréteur graphique
- 5.1 Description des commandes
  - 5.1.1 Rotation d'un bras
  - 5.1.2 Pivotage d'un bras
  - 5.1.3 Translation du robot
  - 5.1.4 Sauvetage d'une position intermédiare
  - 5.1.5 Retour à la position intiale
  - 5.1.6 Retour à la position intermédiaire
  - 5.1.7 Affichage des coordonnées, des points de la trajectoire, à l'écran
  - 5.1.8 Affichage des coordonnées, des points de la trajectoire, sur imprimante
- 5.2 Communication entre programmes
  - 5.2.1 Introduction
  - 5.2.2 Communication : librairie de procédures et fonctions
    - 5.2.2.1 xx_initialisation
    - 5.2.2.2 xx_made_robot
    - 5.2.2.3 xx_end
    - 5.2.2.4 xx_send_command
    - 5.2.2.5 xx_get_info
  - 5.2.3 Changement de point de vue
  - 5.2.4 Que se passe - t-il en coulisse
    - 5.2.4.1 xx_initialisation
    - 5.2.4.2 xx_made_robot
    - 5.2.4.3 xx_send_command
    - 5.2.4.4 xx_get_info
  - 5.2.5 Avantage d'utiliser l'interface de communication

6 Conclusion

7 Propositions d'adaptation

- 7.1 Construction du robot

7.2 Visualisation du robot

7.3 Interprétation sémantique des commandes
7.4 Interprétation des résultats

7.5 Robot ayant plusieurs points fixes

7.6 Robot ayant plusieurs organes actifs

PARTIE 2

8. Lancement du programme
8.1 Installation sur disquette

8.2 Installation sur disque dur

8.3 Contenu du disque de distribution
8.4 Lancement du programme

8.5 Restrictions
9 Manuel détaillé d'utilisation
9.1 Saisie et correction des données
9.1.1 Saisie et correction d'un caractère
9.1.2 Saisie et correction d'une suite de caractères
9.1.3 Sélection d'un champ sans saisir
9.2 Construction du robot
9.2.1 Lecture d'un robot existan
9.2.2 Création d'un nouveau robot
9.2.2.1 Ajout d'un bras
9.2.2.2 Modification des caractéristiques d'un bras
9.2.2.3 Suppression d'un bras
9.2.2.4 Visualisation de la structure du robot
9.3 Déplacement du point fixe

9.4 Sauvetage du robot

9.5 Configuration du système
9.5.1 Configuration hardware
9.5.2 Carte graphique
9.5.3 Choix de la langue
9.5.4 Choix de la couleur
9.5.5 Sous-directory par défaut
9.5.6 Sauvetage de la trajectoire
9.5.7 Choix relatif au angle A et B

PARTIE 3

Annexe 1 : architecture logique

Annexe 2: architecture physique

Annexe 3: spécifications

Chapitre 0

Introduction

0. INTRODUCTION

Le logiciel développé dans le cadre de ce mémoire de fin d'étude a pour but de faire la simulation du comportement d'un robot soumis à un certain nombre de commandes données par un utilisateur via le terminal écran ou un fichier de commandes.

Le robot est défini par un certain nombre de caractéristiques particulières ayant rapport à ses éléments constitutifs (nombre de bras, objet de liaison entre deux bras,...). Les robots modélisables dans l'état actuel du logiciel sont constitués d'une suite de bras terminée par un organe actif. Ces robots n'ont qu'un seul point en contact avec la surface d'appui.

La priorité ayant été donnéeà la qualité de la simulation, le logiciel a dû être limité à cette catégorie de robot. En effet, le traitement de robots plus développé aurait augmenté très fort le degré de complexité. Il nous a dès lors semblé préférable de limiter le logiciel à des robots relativement simples pour avoir une simulation de meilleure qualité. Ce logiciel pourra, cependant, aisément être étendu à un ensemble de robots plus complexe en apportant les améliorations proposées à la fin de cet écrit.

Le logiciel développé est destiné à faciliter le travail des ingénieurs en robotique qui sont chargés d'élaborer les programmes de commandes à faire exécuter par un robot particulier.

En effet, un robot est construit dans le but d'exécuter un certain nombre de tâches de façon automatique et répétitive. Pour ce faire, il est nécessaire de leur transmettre de quelque manière que ce soit (manuelle ou programmée) la suite de commandes élémentaires à exécuter pour effectuer une tâche donnée.

Cependant, les ingénieurs disposent rarement du robot et des conditions suffisantes lors de la conceptions de ces programme. Il leur est dès lors difficile de pouvoir, pour chaque tâche, penser en terme d'angles de rotation, d'ampleur de mouvement,... et ce pour chaque composant du robot.

En vue de faciliter le travail de ces ingénieurs, il peut être intéressant de disposer d'un logiciel qui a pour but de simuler le comportement d'un robot soumis à un certain nombre de commandes. Cela doit pouvoir leur permettre de trouver plus rapidement la suite des commandes nécessaires à l'exécution complète d'une tâche donnée.

Le logiciel développé permet, dans un premier temps, de construire tout en le visualisant à l'écran et ensuite de lire et interpréter des commandes destinées à ce robot.

Ces commandes sont fournies à partir d'un autre programme écrit dans un language quelconque (Pascal,C,Basic,...) par un utilisateur particulier. La communication entre le programme de simulation et celui de lecture des commandes se fait à l'aide d'un certain nombre d'outils mis à la disposition de l'utilisateur.

Afin que le logiciel de simulation soit d'une grande utilité lors de la conception du programme d'un robot, il est intéressant pour l'utilisateur de pouvoir tester si la suite des commandes correspond au mouvement qu'il désirait faire exécuter par le robot. Pour ce faire, le logiciel lui offre deux possibilités afin de pouvoir faire la vérification voulue.

La première consiste à présenter la trajectoire par laquelle passe l'extrémité du robot. Cette présentation peut se faire de plusieurs manières : affichage des coordonnées des points, visualisation de la trajectoire à l'écran par connection des points par des segments de droites,...

La deuxième possibilité offerte à l'utilisateur consiste à lui permettre de voir le robot selon tous les points de vue possibles. Cela consiste à lui permettre de pouvoir "tourner" autour du robot afin de le voir selon tous les angles de vision.

Afin de donner à l'utilisateur la possibilité de mieux se rendre compte de la réalité représentée à l'écran, la visualisation des bras du robot se fait en tenant compte de la distance entre chaque point du robot et l'observateur extérieur.

Le langage utilisé pour la réalisation de ce logiciel est le TURBO PASCAL version 5.0. Ce langage est suffisamment puissant pour permettre l'exécution axec des performances suffisamment bonnes.

Pour ce qui est de la représentation graphique du robot, les outils fournis par le langage permettent sa réalisation sans ralentir l'affichage de façon trop visible. De plus ce langage permet de représenter tous les éléments constitutifs d'un robot de manière bien distincte.

En ce qui concerne les calculs nécessaires à la spécification de la position résultant d'un mouvement particulier, il n'était pas nécessaire d'utiliser un langage plus proche du système d'exploitation tel que C, la partie calcul étant relativement courte pour chaque mouvement. De plus TURBO PASCAL version 5.0 possède une option de compilation qui permet d'exploiter les avantages d'un co-processeur mathématique dont serait muni le micro-ordinateur utilisé. Cela permettrait d'accélérer la partie calcul.

La première partie de cet écrit consiste en une description de tous les éléments intervenant dans la réalisation de ce logiciel. Elle commence par une présentation de ce qu'est la simulation. Ensuite, une présentation de la robotique permet de mettre en évidence l'utilité du logiciel développé. Le chapitre suivant consiste en une description de tous les concepts relatifs au robot modélisé. Après cette description, il est utile de présenter l'interface de ce logiciel avec le programme chargé de définir les commandes à faire exécuter. Enfin , après avoir expliqué le caractère résident du logiciel et décrit les zones de communication entre le logiciel et le programme auxiliaire, il reste à présenter une série d'améliorations à apporter au logiciel dans le but de traiter des robots plus complexes.

La deuxième partie comprend un manuel d'utilisation du logiciel contenant une description de la marche à suivre pour lancer le programme et pour le faire exécuter correctement.

Les annexes constituent une troisième partie. Elles sont composées de l'architecture logique et physique ainsi que des spécifications complètes des différentes procédures et fonctions nécessaires pour l'implémentation du logiciel.

Chapitre 1

Simulation

1. SIMULATION QUANTITATIVE

1.1 DEFINITION DE LA SIMULATION

Une signification précise et universelle du mot simulation n'existe pas. Sa signification change souvent en fonction du domaine d'application. Une raison est que, dans la plupart des cas, la simulation est l'instrument choisi par de nombreuses personnes pour faire face à des problèmes d'une complexité certaine. Sa signification est précisée par ses utilisateurs en fonction de la manière dont il l'emploie. Pritsker, (1979), présente une longue liste de définitions de la simulation. En voici quelques unes :

-" La simulation dénote une technique numérique, basée sur l'ordinateur, pour des études expérimentales d'un processus stochastique ou déterministe dans le temps."

-" La simulation est la technique de construction et d'exécution d'un modèle du système réel afin d'étudier le comportement de ce système, sans modifier l'environnement du système réel. "

- "La phrase "modèlisation et simulation" désigne l'activité complexe associée avec la construction de modèles des systèmes réels et leur simulation sur un ordinateur. En particulier, la modèlisation s'occupe d'établir des relations entre les systèmes réels et les modèles; la simulation essaie, quant à elle, de définir les liens entre les modèles et les ordinateurs."

- "La simulation est l'établissement d'un modèle mathématique d'un système et sa manipulation expérimentale sur un ordinateur digital.

Nous définissons la simulation d'un système comme étant la résolution de problèmes en suivant les modifications, dans le temps, d'un modèle dynamique d'un système."

Etant confronté au difficile problème du choix, nous allons adopter une définition largement utilisée qui généralise les précédentes: (Oren, 1981).

La simulation est l'activité d'expérimentation à l'aide de modèle.

Nous ne sommes pas les seules à adopter cette défintion. Selon le "Dictionnaire de l'Informatique Larousse", 6ème édition, il est dit:

"Ensemble de techniques permettant d'étudier le comportement futur d'un système à partir d'un modèle mathématique approprié, programmé de manière à étudier l'évolution des différentes variables représentatives du phénomène que l'on cherche à analyser. Les techniques de simulation permettent de prévoir le comportement de systèmes physiques complexes (ponts, avions, fusées, centrales nucléaires, etc.) ou théoriques (programme, équations de la physique, modèles économiques, etc.) sans qu'il soit nécessaire de disposer, dans un premier temps, des systèmes réels. Ainsi en informatique peut-on étudier le comportement d'un logiciel destiné à un type futur d'ordinateur sur un ordinateur actuel, grâce à l'utilisation d'un programme de simulation".

Nous pouvons remarquer que :

- La simulation est une notion indépendante du domaine auquel elle s'applique; une fois que le domaine d'expérimentation est défini (ex : file d'attente sur un réseau, modèle économétrique , physiologie humaine, simulateur de vol, ...), les outils disponibles par la technique de simulation sont offerts pour réaliser des études de systèmes réels dans le domaine concerné.

La simulation est une théorie appliquée au système. Mais pour combler le fossé existant entre le monde réel et le monde des modèles, de nouveaux problèmes sont crées :

- choix d'un modèle : structure du modèle, valeurs des paramètres, bonnes simplifications.
- domaine de validité de chaque modèle.
- les familles de modèles semblables peuvent être traitées par des méthodes similaires. Un exemple de modèle analogue est l'utilisation de la théorie des files pour modéliser la circulation capillaire de sang : chaque globule de sang est considéré comme un client et le système capillaire est le serveur.

1.2 DEFINITION D'UN MODELE

Les théories sont presques toujours exprimées dans la forme d'un modèle. L'équation F = k.m1.m2/d (la force entre deux corps est proportionnelle

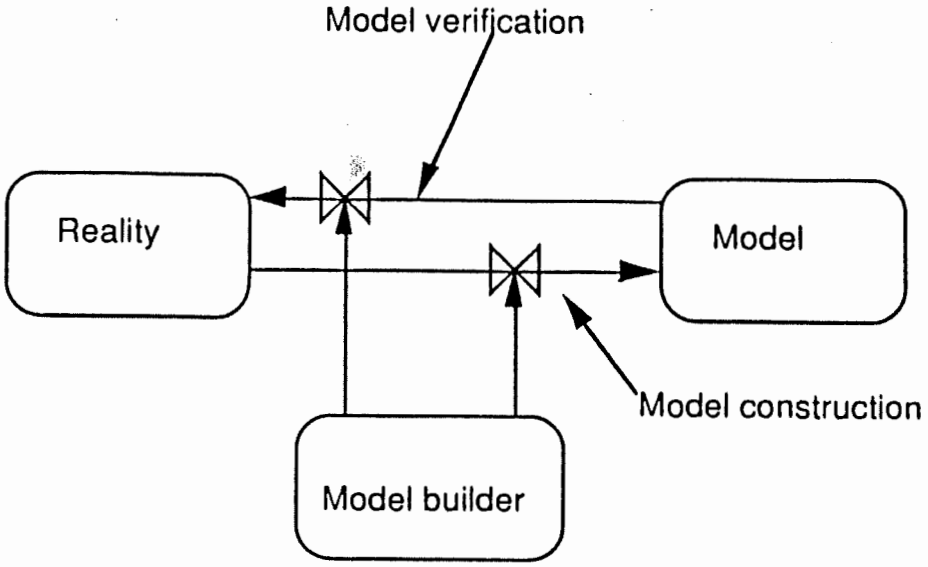
au produit des masses et de l'inverse du carré de la distance) est un exemple de modèle, un modèle mathématique. Fréquemment, le modèle est constitué par un ensemble d'équations mathématiques qui réalise les concepts fondamentaux d'une théorie. Des modèles physiques sont parfois utilisés, où les équations du modèle mathématique sont matérialisées par un système d'une nature différente. La prédiction du temps, les circuits électriques et les ordinateurs analogiques en sont quelques exemples. Le modèle d'un ordinateur analogique est le circuit électronique d'un ordinateur dont la configuration peut être changée pour être en accord avec le système qui va être étudié. Dans d'autres cas, le modèle est conceptuel. Les équations mathèmatiques sont trop pauvres pour les représenter. Considérons quelques exemples : modèle de BD conceptuel (hiérachique, relationnelle, ...), et les modèles de connaissance (trouvé dans les systèmes experts comme MYCIN, PROSPECTOR,etc.). En résumé, nous pouvons dire qu'un modèle est la représentation de la connaissance que l'expérimentateur possède (ou souhaite posséder) sur le sujet étudié. L'expérimentation à l'aide d'un modèle (simulation) est principalement fait pour augmenter la connaissance sur le système à l'étude. Une simulation doit répondre (et parfois créer) aux questions.

Quand le modèle est présenté dans la forme d'équations mathématiques, nous disons qu'il est dans une forme explicite, sinon, il est dans une forme implicite.

Fondamentalement, les modèles sont utilisés pour mettre une hypothèse dans une forme concise, obligeant le chercheur à spécifier précisèment sa ou ses suppositions. La grande majorité des gens qui ont, au moins une fois, réalisé une expérience de simulation connaisse ce fait. Ils partent d'un problème à étudier et sélectionnent un nombre d'attributs importants pour construire le modèle. Cependant, très souvent, les premières expériences avec le modèle présentent un comportement différent de celui du système réel et ce, dans des aspects importants. De nouveaux attributs pertinents ou des modifications dans les relations entre eux sont introduits pour faire coïncider le comportement du modèle avec celui du système réel. Ce processus est représentée par la figure de la page suivante où l'expérimentateur interagit avec un ordinateur pour construire le modèle.

Une fois que le modèle mathématique a été établi, il peut être utilisé pour quantifier les effets suite au changement de valeur des paramètres, souvent avec des résultats non prévus (principalement dans des cas non linéaires). A ce point, il est tentant de croire que les résultats de la simulation révèlent quelque chose de sûre au sujet de la réalité physique, mais ils révèlent uniquement des choses en fonction des hypothèses adoptées dans le modèle. Une discordance entre le comportement du système et le modèle peut être considérée comme à la première étape. Ce désaccord est à utiliser positivement; il doit permettre d 'établir une théorie plus proche de la réalité, suggérant de nouvelles expériences et une profonde réflection sur le modèle créé. Les modèles sont donc des entités dyna-

miques, suscitant des questions pour de nouvelles expériences, et aidant au développement d'une théorie. Dans un certain sens, cela permet d'augmenter la connaissance sur le système par la construction de modèle.



Remarque : la signification du mot "modèle" utilisé dans ce chapitre est seulement une parmi d'autres. Plusieurs significations différentes peuvent être trouvées :

- modèle d'un avion
- modèle d'une voiture
- citoyen modèle.

Le modèle d'un avion est un système simplifié qui reproduit certaines caractéristiques d'un avion du monde réel ou d'un monde plausible (par ex. un monde similaire au notre ou un mode de science fiction). Le modèle peut être exprimé comme un ensemble d'équations mathématiques (modèle mathématique), comme un autre objet avec une forme générale d'avion mais d'autres dimensions (modèle réduit), ... (**simulation meaning**).

Un modèle de voiture est un type de voiture, normalement sélectionné parmi d'autres offerts par le constructeur (**selection meaning**)

Un citoyen modèle est un citoyen idéal ou standard pour l'évaluation des autres, les citoyens les moins parfaits; il définit un objectif recherché et peut existé seulement dans un monde non plausible (**prototype meaning**).

1.3 BUTS D'UNE ETUDE DE SIMULATION

Il y a différentes motivations possibles pour entreprendre une étude de simulation.

Sujets de recherche : Les modèles sont utiles pour augmenter notre compréhension d'un système réel. La construction du modèle accroît parfois notre connaissance et le modèle peut remplacer le monde réel dans les expérimentations. Un exemple de tel modèle est la loi de COULOMB d'attraction et de repulsion entre deux charges électriques; en fait, ce n'est pas une loi dans un sens "légal"(qui est établi et qu'il faut respecter), c'est un modèle du monde physique. Cependant, comme les conclusions sont basées sur les résultats des expériences sur le modèle, une attention toute particulière doit être portée à l'étude des limites du modèle. Les critéres pour établir l'acceptabilité des résultats d'expérimentations utilisant le modèle sont, probablement, cruciaux, difficiles à mettre en évidence, non généraux, et parfois inconnus, mais extrémement importants. Cela représente très souvent un point critique : le modèle doit être suffisament précis, mais les particularités qui ne sont pas présentes dans la réalité ne doivent pas l'être dans le modèle.

Conception assistée par ordinateur (CAD) : dans la conception d'un système complexe, la simulation peut être utilisée pour prévoir son comportement (conception de prototypes d'avion).

Aide à la décision : la simulation permet de susciter différentes politiques de décision qui pourraient aider les actions à accomplir. Le diagnostic médical est un exemple, où le modèle aide le médecin à choisir le traitement qui doit être suivi. Différentes représentations peuvent être possible pour ce modèle. Le modèle peut être présenté comme un système d'équations mathématiques. MAXPUFF (Dickinson) est utilisé pour l'enseignement de diagnostic de maladies pulmonaires est un exemple. Le programme MYCIN est utilisé pour le diagnostic de meningites est un exemple.

Sujets éducatifs : il est possible en CAI (Computer Aided Instruction) et en ICAI (Intelligent Computer Aided Instruction) d'utiliser un modèle pour expérimenter un sujet. Les principaux avantages sont :

expérimentations moins chères, reproduction des résultats (la variabilité naturelle peut être éliminée) et une grande flexibilité d'hypothèses (situations réalistes et non réalistes peuvent être créés facilement).

Cependant, si les limites de validité du modèle sont inconnues, il y a danger pour l'étudiant de prendre le modèle pour la réalité.

1.4 MODELISATION QUALITATIVE ET QUANTITATIVE

La simulation d'un système physique comprend la construction d'un modèle, généralement dans la forme d'équations mathématiques, la manipulation de ce modèle pour obtenir la solution des équations, et finalement l'interprétation des résultats. Dans une simulation qualitative, nous allons directement du système réel à l'interprétation sans utiliser le modèle mathématique. Cette approche fut introduite récemment par Hayes (Hayes, 79), et a reçu une attention sans cesse croissante de la part de la communité de l'intelligence artificielle qui essaie de reproduire le raisonnement humain pour expliquer le comportement réel de l'intelligence humaine.

La modélisation quantitative souffre de certaines faiblesses, principalement dans les sciences appellées sciences douces (sciences humaines, sociales, etc.) dans différents degrés. Les exemples de difficultés qu'il est possible de rencontrer sont:

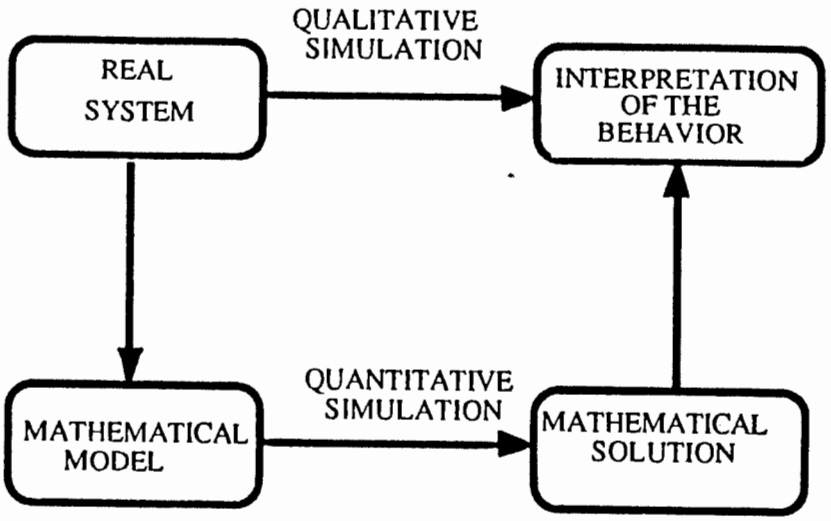
- choix difficile de la structure du modèle mathématique : comment un choix différent de structure influence l'interprétation des résultats ?

- dans les sciences douces, différentes hypothèses sont souvent possibles. Il en résulte différents modèles (Noordergraaf, 80). Quel est celui qui doit être utilisé pour donner le genre d'interprétation désirée ?

- c'est souvent difficile d'avoir une interprétation pour tous les paramètres utilisés dans le modèle.

- la variabilité des paramètres pour différents sujets peut être extêmement large; il est difficile d'évaluer comment cette variabilité est propagée vers les résultats de la simulation.

Dans de tels cas, il est tentant d'utiliser une approche différente, plus directe. La figure ci-dessous montre une telle option.



Pour aller d'une connaissance imprécise sur le système vers une explication de son comportement, au lieu de passer par l'utilisation d'un modèle mathématique, nous avons besoin de représenter notre connaissance sur le système. Cette connaissance comprend la description du système et le mécanisme de raisonnement pour permettre de déduire les résultats désirés de l'expérience, de la simulation. Dans ce cas, les résultats peuvent être directement dans la forme de l'interprétation désirée. Etant donné que les paramètres, les relations et les mécanismes de raisonnement, dans ce cas, sont exprimés sans utiliser de nombres, nous appelons cela de la SIMULATION QUALITATIVE.

1.5 SIMULATEUR DE ROBOT ET SIMULATION QUANTITATIVE

La modélisation du robot effectuée dans notre programme correspond à un ensemble d'equations et transformations; autrement dit, un modèle

mathématique. Ses différents paramètres sont les informations nécessaires pour simuler le mouvement d'un bras, du robot, ...

la nature du mouvement

l'angle ou l'axe concerné par le mouvement

l'ampleur du mouvement

le signe

le numéro du bras

L'interprétation des résultats se fait graphiquement(il est possible de suivre le mouvement à l'écran de l'ordinateur) suite à l'attribution de valeurs aux différents paramètres.

1.6SIMULATION ET GRAPHISME

Des ordinateurs graphiques ont été développés depuis le début de années 80 jusqu'à un point tel que ceux qui sont intéressés par la simulation ont vu la possibilité d'en tirer un maximun. L'apparition de cet outil, capable de produire des images de haute qualité, améliorent sensiblement la qualité et la réputation de la simulation en produisantde plus en plus des systèmes que les non spécialistes peuvent immédiatement utiliser. L'aide fournie aux "simulationistes" se présente de deux manières différentes :

- l'amélioration de la présentation des résultats d'une simulation. Un utilisateur peut plus facilement interpréter les résultats s'ils sont affichés graphiquement.
- l'évolution de l'interface avec un programme de simulation vers un interface essentiellement graphique.

La conception d'un simulateur associé à un graphisme de haute qualité ne s'effectue pas sans devoir surmonter quelques difficultés. Les deux principaux freins dans le développement d'un simulateur dont le graphisme est de haute qualité sont :

- le temps pris pour faire des opérations en virgule flottante, comme des multiplications de matrices, ...

- le temps pris pour afficher les objets.

L'affichage d'une image peut prendre de quelques minutes à quelques heures (temps d'exécution). Dans un film vidéo, l'effet du continu est assuré par la production de 30 images par seconde. S'il fallait réaliser une simulation de quelques minutes, comme c'est le cas pour les dessins animés, cela pourrait demander des semaines de travail de préparation. Une simulation accompagnée d'un graphisme de haute qualité est à l'heure actuelle impossible à réaliser, même avec la vitesse atteinte par les supers ordinateurs. En effet, quelque soit la méthode utilisée pour représenter les objets dans une situation assez réaliste, un millier de données sont requises; chacune d'elle étant employée comme opérante dans un nombre d'opérations avant que la composante graphique qui y correspond ne soit affichée. Même si ces opérations se faisaient dans un temps ne ralentissant pas l'animation, le processus d'affichage devrait être assez rapide pour assurer un défilement de 30 images par seconde.

Il est évident qu'un mono-processeur ne saurait assurer, à lui seul, une vitesse d'exécution telle que 30 images de haute qualité soient produites. De nombreuses recherches sont en cours pour produire du hardware avec une architecture en parallèle. Cette solution ne résoudrait qu'un seul des deux problèmes. Le temps de calcul serait réduit par la présence de plusieurs processeurs ainsi que d'algorithmes de calcul travaillant sur un ensemble de données partagées judicieusement. Le frein dans l'animation de graphisme de haute qualité reste le processus d'affichage. L'affichage d'images de haute qualité demande le respect d'un certain nombre d'aspects :

- projecter les objets en perspective et enlever les surfaces cachées
- tenir compte des ombres provoquées par les objets
- tenir compte de la réflexions et réfractions au travers d'objets transparents
- ...

A l'heure actuelle, l'animation de graphiques en 3D est possible mais ne bénéficie pas des finesses de l'affichage (ombre, ...). Ces différents raffinements ralentissent d'une manière relativement importante l'affichage.

De par la nature de son processeur (mono-processeur), la simulation de graphiques de haute qualité sur IBM PC est très limitée. Le prix à payer est une animation quasiment nulle. Il suffit, pour s'en rendre compte, d'exécuter des softwares présents sur le marché, tel AUTOCAD.

Chapitre 2

Robotique

2. PRESENTATION DE LA ROBOTIQUE

2.1. Introduction

Notre mémoire touche le domaine de la robotique. Par ce chapitre, nous essayons de relever les éléments qu'il est pertinent de connaître pour comprendre les notions utilisées dans le simulateur de robot. Nous allons tenter de "définir" un robot, voir quels en sont ses composants. Ensuite, nous examinerons les différents modes de programmation ainsi que les outils généralement utilisés pour programmer un robot.

2.2 Qu'est-ce qu'un robot : première approche.

* un robot est une machine qui peut facilement être utilisée pour accomplir une variété de tâches tout en se passant d'une supervision humaine.

* les robots industriels typiques sont des machines immobiles, habituellement boulonnées au sol ou dans une position aérienne, qui peuvent saisir des objets et les déplacer. Les robots industriels sont généralement employés pour faire des travaux dangereux et difficiles pour un être humain.

* les robots équipés d'un "cerveau" sont en phase de développement. Ils pourront voir, parler, reconnaître la voix et avoir un sens du toucher et auront la possibilité de réagir de différentes manières en fonction de l'évolution de son environnement.

* certains robots sont mobiles et peuvent suivre un chemin préprogrammé. D'autres robots, parfois appelés des robots personnels, peuvent se "promener" dans une pièce ou une maison et apprendre à ne pas heurter les meubles ou murs. Ces robots peuvent apparaître être intelligents même s'ils suivent un ensemble complexe d'instructions programmées. Etant donné que les robots deviennent plus sophistiqués, les robots sembleront être plus "intelligent". Le développement de l'intelligence pour les robots résultent du développement de l'intelligence artificielle en informatique.

Voilà quelques idées de ce que représente un robot. Il n'est pas question de faire prévaloir l'une par rapport à l'autre. Une certitude subsiste cependant : le robot tend à remplacer, dans un avenir plus ou moins proche, l'homme dans de nombreux domaines. Il est possible de s'en rendre compte en regardant le nombre de robots progressivement installés dans différents pays.

Nombre de robots installés par pays de 1985 à 1990

Pays	1985	1990
JAPON	16.000	29.000
ETATS UNIS	7.700	31.000
RFA	5.000	12.000
SUISSE	600	5.000
SUEDE	2.300	5.000
NORV7GE	1.000	2.000
GRANDE BRETAGNE	3.000	21.000
POLOGNE	200	1.400
DANEMARK	110	250
FINLANDE	950	3.000
BELGIQUE	200	—
YOUGOSLAVIE	150	300

"Fundamentals of Robotics

Theory and Applications

Larry Heath
Indiana State University"

2.3 Robot: Définition

Aucune définition d'un robot n'est universellement acceptée. Ainsi, on peutretrouver différentes définitions.

WEBSTER : (Webster's seventh new Collegiate Dictionary, Third Edition G & C. Herriam Company, Publishers, Spengfield, Massachusetts, 1963)

1. Une machine de forme humaine qui réalise des fonctions mécaniques d'un être humain mais manque de sensibilité.

2. Un outil ou appareil automatique qui réalise des fonctions ordinairement attribuées à un être humain.

3. Un mécanisme guidé par des contrôles automatiques.

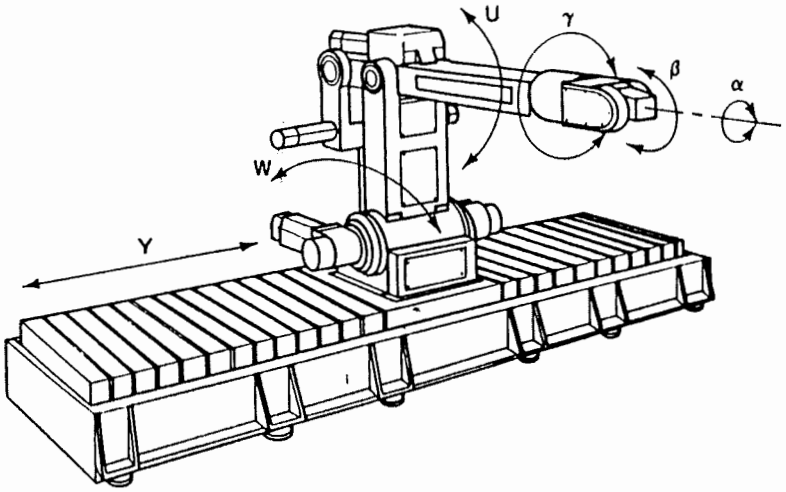
Nous avons décidé de prendre la définition suivante en ce qui nous concerne vu la notion de robot utilisée dans le simulateur: (The Roboties Institute of America) *un robot est un manipulateur reprogrammable, multifonctionnel destiné à bouger des matériaux, outils, des objets spécialisés et ce, au travers de variables "programmées" qui influencent le mouvement.*

2.4 Les composants d'un robot

Un robot est une machine complexe composé de plusieurs parties importantes. Chacune de ces parties doit être conçue pour travailler avec les autres parties du système aussi bien qu'avec les autres machines dans l'usine. Les parties principales d'un robot sont le bras, le poignet, la main et l'ordinateur. Pour ce qui nous concerne, le poignet et la main forment l'organe actif.

2.4.1 Bras du robot

La partie la plus évidente est le bras (le bras est ici compris dans le sens où il peut être composé de plusieurs éléments, c'est en fait la partie entre la base du robot et l'organe actif). Il est généralement conçu pour déplacer un objet à l'intérieur d'une aire donnée, et pour soulever un poids maximum. Les configurations communes des bras d'un robot permettent une quantité de mouvements. Le plus simple des robots serait une machine à deux axes. (Cela déplacerait simplement les parties de haut en bas, par exemple). Le terme "axe", quand il est utilisé en prenant comme référence le robot, stipule le nombre de manières indépendantes suivant lesquelles le robot peut se mouvoir. Cela est aussi appelé les degrés de liberté du robot. Les robots sont construits de 2 à 10 axes, ou degrés de liberté. La plupart des robots ont 5 à 6 degrés de liberté. Les degrés de liberté sont illustrés par la figure page suivante.



2.4.2 L'organe actif

Le bras d'un robot est terminé par ce que l'on appelle un organe actif. Cet organe actif permet de saisir, d'accomplir certaines choses... Bien qu'un robot présente un caractère de flexibilité et d'adaptabilité, l'organe actif est généralement spécifique à la tâche qu'il doit réaliser. Ainsi, une pince conçue pour manipuler des portières de voiture ne sera pas utilisée pour saisir un siège ou un tableau de bord.

2.5 Programmation d'un robot

2.5.1 Introduction

La clé du succès et de l'utilité des robots est leurs capacités à suivre successivement une séquence de commandes qu'ils ont apprise. Ce chapitre essaie de décrire comment le robot se déplace, comment il interagit avec d'autres machines, et comment il apprend à suivre un chemin précis. Autrement dit, comment il est programmé. Une analogie utile peut être faite entre la programmation d'un robot et la conduite d'une automobile. Sans le "programmeur", le robot ne

sait rien faire. Une voiture sans chauffeur n'est d'aucun usage. Les bases de la programmation sont très simples mais pour réaliser ce travail correctement, cela requiert une attention toute particulière à certains détails ainsi que la connaissance du but poursuivi.

Il existe deux manières de programmer un robot. Il y a une programmation on line et une programmation off line. Mais quelque soit le travail que robot doit accomplir, le robot réalise une séquence d'étapes. Voyons d'abord un exemple où le travail est réalisé par un robot.

2.5.2 Travail réalisé par un robot : exemple.

Dans le processus de chargement de grosses pièces sur une palette à partir d'un tapis roulant, il serait typique d'employer un robot. Monsieur Joseph ENGELBERGEN décrit ce type de processus dans son livre "Robot is in practice" quand il décrit, étape par étape, comment apprendre à un robot à décharger un cylindre d'un tapis roulant vers une palette.

Il est possible, par ce simple exemple, de se rendre compte que la programmation d'un robot n'est pas une chose banale; le programmeur étant amené à penser à toutes les étapes d'un mouvement.

1. Déplacer le bras du robot de sorte que l'organe actif, une pince dans notre cas, soit juste au-dessus du cylindre désiré, tout en gardant la pince ouverte.

2. Faire tourner l'organe actif et le poignet pour aligner la pince dans un plan horizontal

3. Enregistrer cette position intermédiaire.

4. Descendre le bras afin que la pince entoure le cylindre.

5. Enregistrer cette position intermédiaire.

6. Fermer la main, de sorte que le cylindre soit saisi.

7. Enregistrer cette position intermédiaire.

8. Lever le bras bien au-dessus du tapis roulant et bien au-dessus de n'importe quels obstacles potentiels qui pourraient exister entre le tapis roulant et la palette, lieu de destination du cylindre.

9. Enregistrer cette position intermédiaire.

10. Faire pivoter le robot, dans son entierté, et ajuster l'extension du bras pour placer approximativement le cylindre au-dessus de la première pointe, de la palette, sur laquelle viendra s'enchevêtrer le cylindre.

11. Enregistrer cette position intermédiaire.

12. Descendre le bras du robot afin de placer l'orifice du cylindre juste au-dessus de la première pointe de la palette, sans la toucher. Assurez-vous que le cylindre est tenu verticalement, et si nécessaire, ajuster l'organe actif, le poignet et le bras pour réaliser cet état.

13. Enregistrer cette position intermédiaire.

14. Descendre le bras jusqu'au moment où le cylindre s'enchevêtre sur la pointe de la palette.

15. Enregistrer cette position intermédiaire.

16. Ouvrir la pince.

17. Enregistrer cette position intermédiaire.

18. Lever le bras du robot à une hauteur certaine pour qu'il soit assuré de ne pas rencontrer d'obstacles.

19. Enregistrer cette position intermédiaire.

20. Répéter cette "opération" pour l'ensemble des cylindres restants, jusqu'au moment où la palette est remplie.

21. Remplacer la palette remplie par une palette vide, commuter le robot dans le mode normal, metter en marche le tapis roulant et le robot, et contrôler si les opérations enregistrées sont celles qui sont désirées. Dans le cas contraire, il est toujours possible d'apporter une modification (cfr infra: le mode edit).

Dans un premier temps, l'exécution de ces différentes tâches se fera on line. Mais par la suite, l'exécution se fera off line étant donné que les commandes sont enregistrées au fur et à mesure de la mise au point du mouvement de chargement de la palette.

Le nombre d'étapes requises pour charger un cylindre sur une palette étant élevé, cela demande la mémorisation d'une grande quantité de données contenant en détail toutes les informations nécessaires pour contrôler un robot, même pour un travail relativement simple (la plupart des robots industriels auront une bande magnétique, un disque magnétique ou une mémoire de masse pour sauver toutes ces informations ....). Toutes ces informations qui forment le programme, ne sont pas facilement à imaginer. Mais une fois que le programme du robot a été développé, il peut être mémorisé et exécuter un nombre infini de fois, chaque fois qu'il est utile de le faire.

2.5.3 Développement du programme pour robot

Les programmes destinés au robot seront développés avec plus ou moins de difficultés. La difficulté est proportionnelle à la difficulté des tâches qui doivent être réalisées avec un robot. Un programme simple, tel que lever le bras du robot un certain nombre de fois, ne présente guère de difficulté. La programmation de ce type de mouvement ne prendra que 3 à 4 minutes. Par contre, un programme modérément complexe, tel que le chargement d'une palette, chargée de pièces, à partir d'un tapis roulant, peut prendre 1 à 2 heures. La principale considération dans une tâche telle que présentée ci-avant est l'interaction avec le monde externe. Existe-t-il une manière automatique pour arrêter et démarrer le tapis roulant ? Une fois les matériaux chargés sur la palette, y-a-t-il une méthode automatique pour enlever la palette ? Comment de nouvelles palettes sont-elles mises en place pour un nouveau chargement ? Résoudre tous ces problèmes et préparer la station de travail de telle manière que l'on puisse progammer le robot est évidemment une entreprise importante. Une fois que tout ce travail préparatif est terminé, le développement et l'édition du programme devraient prendre seulement une ou deux heures. Voyons les étapes principales à suivre lorsqu'on veut introduire un robot pour réaliser une tâche, un travail.

2.5.3.1 Définition du travail

C'est une analyse détaillée de l'ensemble de la tâche. Les principaux éléments de cette définition comprennent l'apport, la fourniture de matériels au robot, une définition précise des tâches que le robot réalisera, et comment les produits finis seront traités, évacués par la suite.

2.5.3.2 Conception de l'organe actif

L'organe actif, à la fin du bras du robot, est un des facteurs critiques dans l'ensemble du job. Si la tâche est relativement simple, un organe actif standard peut être utilisé. Avec des travaux complexes, de nouvelles conceptions peuvent être nécessaires mais si elles sont bien conçues, la réalisation de la tâche, en fin de compte sera d'autant plus simple à garantir.

2.5.3.3 Développement du programme

Il existe plusieurs manières de programmer un robot : la programmation ON - LINE et programmation OFF - LINE. Les méthodes les plus populaires et le plus utilisées jusqu'à présent sont toutes ON - LINE, mais un effort sensible est réalisé pour faire progresser la programmation OFF - LINE qui utilise un langage de programmation.

A) Programmation on - line.

A.1) Programmation par démonstration (teaching by showing)

Une personne qualifée dans la réalisation de la tâche que le robot aura à effectuer saisit l'organe actif du robot et exécute avec lui les mouvements qu'il devra reproduire. Pendant ce temps, le contrôleur du robot mémorise plusieurs fois par seconde la position du robot. Par la suite, il suffit de repasser séquentiellement par tous les points enregistrés afin de reproduire le mouvement désiré.

A.2) Programmation par boîtier d'apprentissage (teach pendant).

Cette technique demande l'emploi d'un appareillage spécial qui permet au programmeur d'employer les différents boutons ou la manette pour faire bouger le robot. Il est ainsi possible, grâce à cette commande à distance, d'amener l'organe actif du robot n'importe où dans l'espace de travail du robot. Le programmeur le conduit aux différentes positions intermédiaires à atteindre et enregistre chacune de celle-ci. Il suffit alors au robot de reproduire la séquence de positions intermédiaires enregistrées.

B) Programmation off-line.

Depuis l'apparition des robots industriels, il y a toujours des langages pour les programmer, même si ces robots étaient prévus pour être programmés par démonstration ou par boîtier de commandes. Malheureusement, ces langages furent, pendant un certain temps, de très bas niveau. Ils offraient généralement un trop petit nombre d'instructions tel l'ouverture et la fermeture de la pince, une intruction qui spécifie une position désirée pour le robot, une instruction spécifiant l'angle de rotation d'un composant du bras, ... Il va sans dire qu'un tel langage était extrêment difficile à utiliser tel quel; aucune visualisation n'étant possible. Il était également difficile de réflichir en terme d'angle de rotation pour chacune des articulations,.... Programmer une tâche était vraiement très complexe. Heureusement, depuis quelques années, des langages de programmations ont été dévelop-

pés, ce qui permet une programmation plus aisée (Langage VAL développé par Unimation pour le robot PUMA, 1982(NAG 84), ...)..

2.5.4 Documentation

Les programmes de robotique, commes les programmes d'ordinateurs, sont passionnants à développer mais cela l'est généralement moins lorsqu'il s'agit de les documenter. Cependant, la rédaction d'une bonne documentaion représente une étape importante. Le développeur de l'application doit définir tout ce qui a été fait et noter tous les changements qui ont été fait depuis les plans initiaux jusqu'à la fin, c'est-à-dire le fonctionnement de l'application.

Quand ces différentes étapes sont suivies avec attention, des applications utiles en robotique peuvent être développées et utilisées. Pour être capable de faire toutes ces étapes, le programmeur doit savoir toutes les possibilités de programmation du robot avec lequel il travaille. Mais avant de s'attarder sur les possibilités de programmation d'un robot (<=> quelles sont les commandes qui peuvent être exécutées par le robot), il est intéressant de remarquer qu'il existe plusieurs manières différentes de programmer un robot. Le lecteur a déjà pu prendre connaissance de deux de ces modes : la programmation on line et off line.

2.5 Différents modes d'opération.

La plupart des robots fonctionnent selon différents modes. Les modes les plus communément rencontrés sont un mode manuel(on line), un mode programme, édit et automatiquement. La programmation off line regroupe le mode automatique avec le mode programme.

* Manuel

Ce mode permet à l'opérateur de démarrer le robot sans avoir au préalable programmé une suite quelconque d'opérations. Dans ce mode, le "teach pendant" est actif. L'opérateur s'en sert pour transmettre, coup par coup, des ordres au robot.

* Programme

Ce mode est utilisé lorsqu'un opérateur décide de programmer un robot. Une fois que le programme est mis au point, il peut être exécuté pas à pas et ce, dans le mode step by step, pour contrôler les opérations programmées. En plus des opérations qu'un robot peut effectuer dans le mode manuel, le program-

meur a à sa disposition un certain nombre de fonctions qu'il peut apporter à chaque étape du programme au robot.

· Delay
Cette fonction permet au robot d'attendre dans sa position courante une certaine période de temps déterminée. Cela peut aller de quelques secondes à quelques minutes. Cela est utile en attendant des opérations qui devraient être exécutées par d'autres machines.

· Wait
Elle est similaire à la fonction DELAY, mais le robot attend un signal ou une combinaison de signaux électriques provenant d'une autre machine avec lequel il serait connecté.
· Output
Cela permet au robot d'envoyer un signal électrique vers un ou plusieurs équipements externes pour les mettre sous tension ou au contraire, pour les éteindre.
· Continue
Continue est une commande qui stipule au robot qu'il ne doit pas stopper, s'arrêter à chaque point, étape du programme. Cela permet au robot d'avoir une certaine vélocité, continuité dans son mouvement. Ce type de mouvement est repris pour des applications tel la peinture, ou la soudure...
· Tool
Ce signal permet de contrôler l'organe actif. Il peut correspondre à la réalisation d'une fonction spéciale par l'organe actif.· ····
·Branch
Cela permet au contrôleur du robot de vérifier une variété de signaux et d'appareils avant de continuer son programme. Cette décision de continuer est une décision logique qui peut inclure plusieurs signaux. Cette fonction auxiliaire permet de vérifier si toutes les conditions requises au bon fonctionnement du robot sont vérifiées.

* Edit

Le programmeur utilise ce mode pour changer un programme. Toutes les variables telles les commandes auxiliaires, la vitesse des opérations, peuvent être changées par cette fonction.

* Automatique

Une fois le programme développé, il est nécessaire d'exécuter le programme. Cela est fait dans le mode automatique. Beaucoup de machines permettent une fonction "pas-à-pas" qui fait partie du mode automatique. Quand cette fonction est appelée, le programme peut fonctionner en suivant normalement les différentes étapes du programme ou inversément, peut fonctionner en marche arrière, refaire les étapes mais dans un ordre inverse. Mais le mode automatique simple ne permet qu'une exécution normale du programme jusqu'au moment où il rencontre un ordre d'arrêt. Il existe habituellement 2 manières de donner l'ordre au robot de stopper.

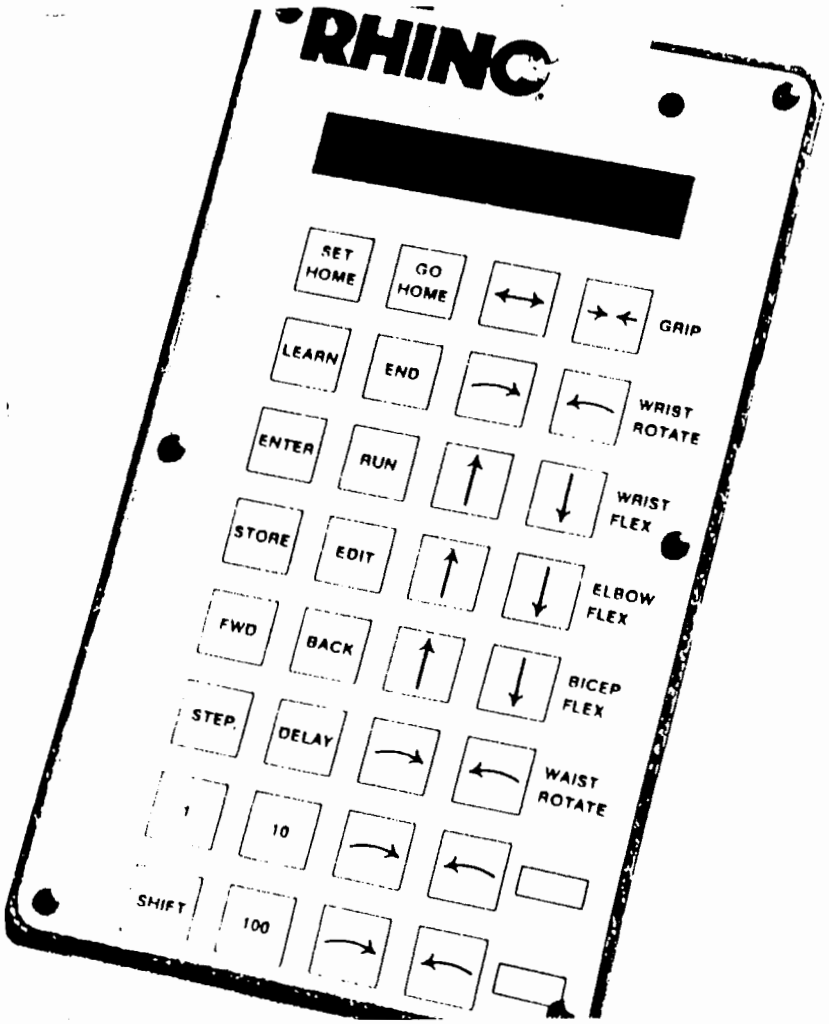
La première est un ordre d'interruption immédiate, la deuxième est un ordre d'arrêt normal lorsque l'exécution du programme arrive à son terme.

Ces 4 modes peuvent être activés à partir du teach pendant. Il existe de nombreux teach pendant. Nous allons d'abord présenter sommairement deux teach pendant existants ainsi que toutes leurs fonctions respectives, ensuite nous considérerons un teach pendant qui serait proche de notre simulateur de robot et les différentes fonctions que il est possible d'exécuter.

2.7 Teach pendant pour le robot Rhino XR-2.

Rhino XR-2 est un robot éducatif. Il est utilisé pour simuler les robots industriels.

2.7.1 Le teach pendant.



Touche : Set home

Fonction : Cette fonction mémorise la position courante du robot, appellée également la position HOME.

Touche : Go home

Fonction : L'appel à cette fonction a pour effet de revenir à la position mémorisée préalablement.

Touche : Learn

Fonction : Le mode d'apprentissage (cfr mode programme ci-avant) est activé.

Touche : End

Fonction : La touche END indique que l'utilisateur a fini de programmer le robot. Si l'utilisateur ne presse pas cette touche et essaie de faire exécuter le programme qu'il a créé, il le perdra. Cette touche, qui est mémorisée dans le programme, arrêtera également l'exécution du robot lorsque le programme sera exécuté.

Touche : Enter

Fonction : Cette touche entre les coordonnées courantes du robot lorsque le mouvement est accompli. Il faut bien entendu se mettre en mode programme. Si cette fonction n'est pas exécutée après le mouvement, ce mouvement ne sera pas mémorisé.

Touche : Run

Fonction : Cette fonction exécute la suite de mouvements qui fut apprise lorsque le mode était le mode Learn (programme).

Touche : Store

Fonction : Le contrôleur du robot peut être attaché avec un computer de marque Appel. Si cette situation est vérifiée, l'exécution de cette fonction a pour effet de stocker le programme mémorisé dans le

teach pendant dans la mémoire du computer. SHIFT · STORE, par contre, charge un programme présent dans la mémoire du computer dans le teach pendant.

Touche : Delay

Fonction : L'exécution de cette fonction provoque une attente d'environ 1.5 secondes.

Touche : 1,10,100
Fonction : Ces touches indiquent l'ampleur du mouvement exécuter en une seule fois.

Touche : Shift

Fonction : SHIFT · END efface le programme mémorisé dans le teach pendant.

L'édition d'un programme existant est une fonction importante. Lorsqu'un programme est mis au point, il l'est pour un certain temps. Il arrive qu'il faille y apporter quelques modifications suite à un changement d'un équipement externe mais en relation avec le robot. Il serait absurde de devoir recommencer l'entierté du programme. L'édition permet au programmeur d'apporter de légères modifications à un programme existant. Les fonctions dont il dispose sont les suivantes :

Touche : Edit

Fonction : L'appel à cette fonction active le mode d'édition.

Touche : FWD

Fonction : L'exécution de cette fonction a pour effet d'exécuter le mouvement suivant du programme et d'afficher les coordonnées.

Touche : Back

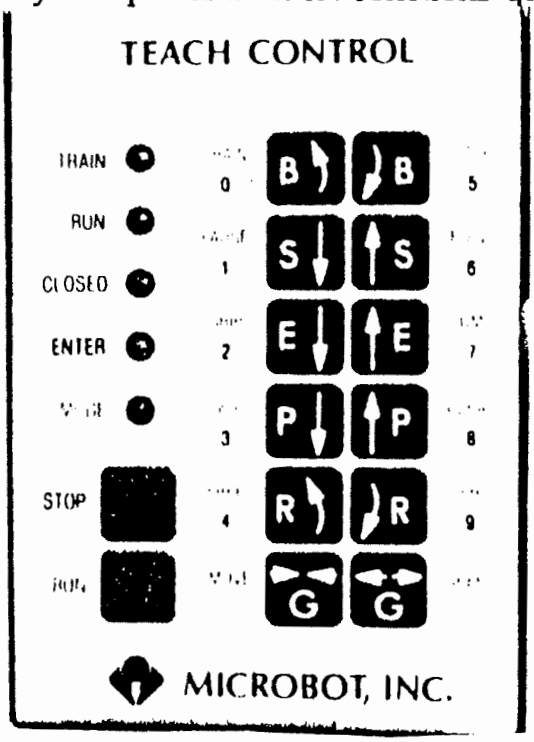
Fonction : L'exécution de cette fonction a pour effet d'exécuter le mouvement précédent du programme mémorisé et d'afficher les coordonnées.

Touche : Step

Fonction : Cette touche est très semblable à FWD. Je n'en dirai pas plus car cela ne nous apprendrait rien sur le genre de fonction qui est offert au programmeur.

2.8 Teach pendant du Microbot teachmover.

Le microbot teachmover fonctionne d'une manière presque similaire aux robots industriels. Voyons quelles sont les fonctions qui sont disponibles sur ce teach pendant.



1. Les 12 touches concernant le mouvement du robot. Ces touches font bouger chacune des parties de la base, du bras, du poignet et de la pince.

2. La touche MODE. Cela permet de choisir entre les fonctions RUN, TRAIN et ENTER.

3. La touche RECORD. Elle est utilisée pour enregistrer la position courante du robot de sorte que le contrôleur du robot, lors de l'exécution, puisse refaire le mouvement (va de positions en positions mémorisées).

4. TRAIN est le mode dans lequel le robot bouge à la suite du jeu avec les 12 fonctions "positionnelles" prévues à cet effet.

5. RUN est la fonction qui permet au robot de répéter une séquence de mouvements (un programme) qui aurait été mémorisée.

6. La touche STOP, lorsqu'elle est appuyée, arrête l'exécution d'un programme. Le robot stoppera instantanément. Si on désire arrêter le robot au point suivant, il faut presser sur la touche REC.

Une fois que le robot est à l'arrêt, une pression sur la touche RUN provoque la reprise de l'exécution à partir du point courant.

7. CLEAR efface tout programme de la mémoire.

8. La fonction ZERO est activée lorsqu'un programme va commencer son exécution. Dans la mémoire du teach pendant, il existe une espèce de pile qui mémorise toutes les positions par lesquelles passe le bras du robot en cours d'exécution. Avant une nouvelle exécution, il est normale de "vider" cette pile. La fonction ZERO l'assure.

9. La fonction PAUSE permet de programmer une attente dans un programme. Il ne faut pas oublier que le robot interagit dans un synchronisme parfait avec d'autres équipements. Le robot doit parfois attendre certaines choses avant de pouvoir continuer son exécution.

10. La fonction SPEED détermine la vitesse à laquelle les mouvements s'exécutent.

11. La fonction STEP provoque l'exécution d'un seul mouvement dans la séquence de mouvements mémorisée. Il est ainsi possible d'exécuter la séquence de mouvements pas à pas en appelant successivement la fonction STEP. Cette fonction est aussi utilisée pour éditer des progammes mémorisés.

12. La fonction GRIIP permet de saisir des objets.

13. La fonction MOVE désactive la fonction REC. Il est ainsi permis à l'utilisateur de bouger le bras vers une nouvelle position de départ sans pour autant vider le contenu de la "pile".

Cela peut être utile lorsque le robot rencontre un obstacle, ... Cette nouvelle position de départ est déterminée à l'aide des touches positionnelles.

14. La fonction FREE est la même que MOVE exception faite que l'opérateur ne déplace plus le bras avec les touches "positionnelles" mais à la main.

15. Il faut simplement retenir que cette fonction OUT permet de contrôler 4 équipements externes, reliés au robot, à partir du teach pendant.

16. La fonction POINT permet de déplacer directement le bras au point numéro X d'un programme mémorisé. Il faut savoir que la mémorisation d'un programme passe par la mémorisation des différents points par lesquels le bras passe.

17. Enfin, la fonction JUMP est un peu spéciale. Son format est le suivant : JUMP X1,X2. X1 est une condition qui doit être testée. Si elle est vérifiée, alors le bras du robot va directement au point X2.

Nous venons d'examiner les fonctions disponibles sur deux teach pendant. Le lecteur peut se rendre compte que les fonctions sont à peu près identiques dans les deux cas.

Il ne faut pas oublier que l'objectif poursuivi au travers de ce mémoire est la réalisation d'un simulateur de robot. Il fallait dès lors énumérer les différentes fonctions, communément présentes sur un teach pendant, que pouvaient réaliser un robot. En résumé, les mouvements demandés à un robot sont les suivants :

- rotation d'un bras(dans le plan horizontal).
- rotation du robot(dans le plan horizontal).
- lever et baisser un bras(dans le plan vertical).
- pivotage d'un bras.
- exécution d'une séquence de mouvements.
- retour à une position préalablement mémorisée.

Remarque importante.

Attention, l'objectif est de réaliser un simulateur de robot et non un simulateur de teach pendant. Les fonctions d'éditions, ... ne sont pas à implémenter.

Regardons à quoi pourraient ressembler le teach pendant du simulateur de robot.

2.9 Teach pendant du simulateur de robot.

En plus des fonctions généralement rencontrées sur les teach pendant, le robot simulé aura la possibilité de se déplacer dans l'espace dans lequel il se trouve, c'est · à · dire selon l'axe X, Y et Z.

Le dessin représentant le teach pendant du robot simulé est présenté à la page suivante.

Les fonctions de ce teach pendant sont les suivantes

1. Il existe 8 fonctions qui sont destinées au mouvement. Le mouvement est relatif à un bras, le bras concerné est celui dont le numéro est ARM NUM. avec une amplitude de ARM STEP si le mouvment n'est pas une translation, sinon TRANSLATION STEP.

2. Set home : Cette fonction mémorise la position courante du robot, appellée également la position HOME.

3. Go home : L'appel à cette fonction a pour effet de revenir à la position mémorisée préalablement.

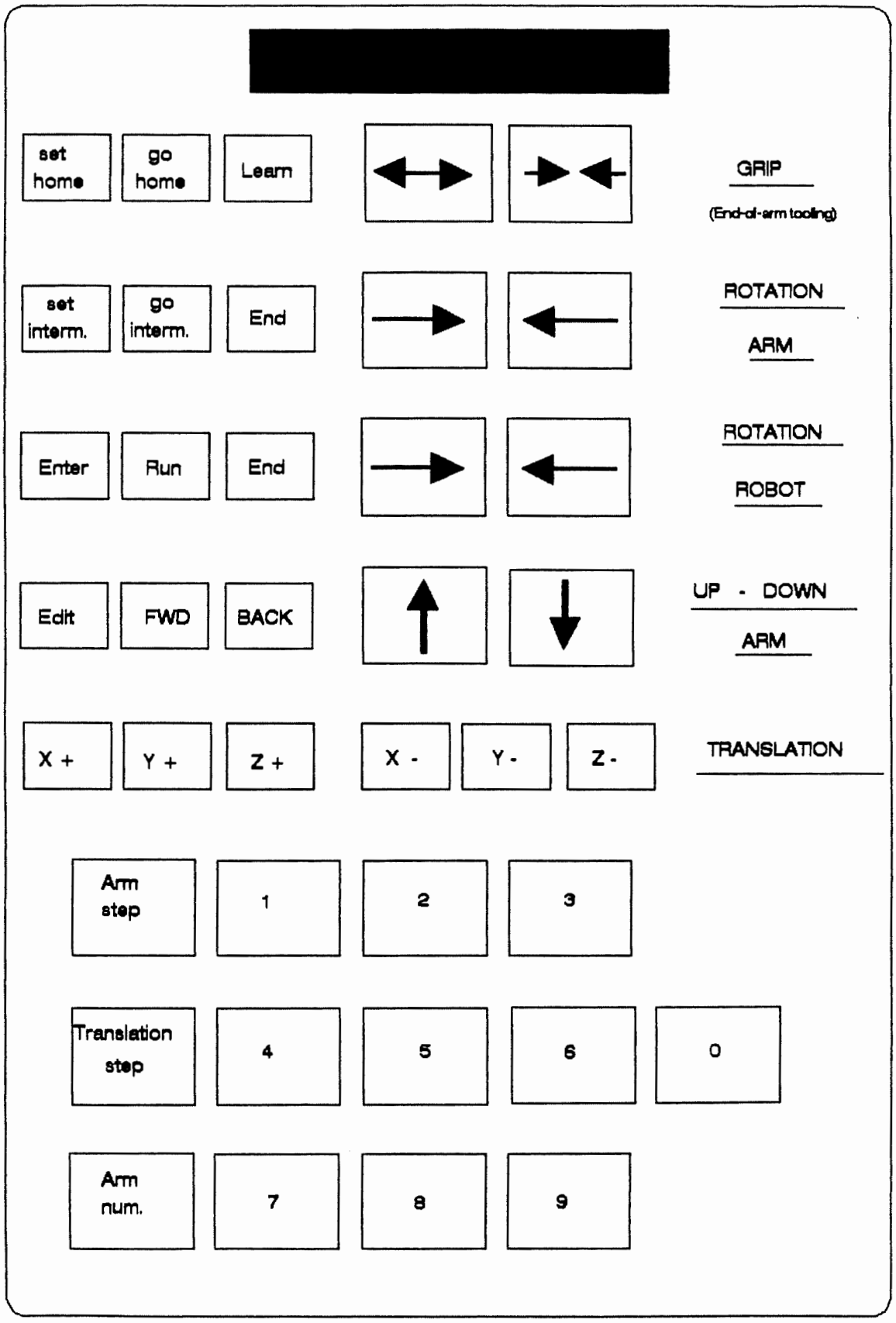
4. Set interm : Cette fonction mémorise la position courante du robot, appellée également la position INTERMédiaire.

5. Go interm : L'appel à cette fonction a pour effet de revenir à la position mémorisée préalablement.

6. Learn : Cette fonction active le mode d'apprentissage. Il est possible de mémoriser une séquence de mouvements pour, par la suite, l'exécuter.

7. End : Cette fonction désactive le mode d'apprentissage (programmation off line). Elle également mémorisée de sorte que l'exécution d'une suite de mouvements stoppe une fois le END rencontré. Le mode d'apprentissage désactivé, c'est le mode de programmation on line qui est actif.

Teach pendant du simulateur de robot.



8. Enter : Cette fonction mémorise les coordonnées courantes lorsque le mode d'apprentissage est actif. Sans exécuter cette fonction , les coordonnées courantes ne sont pas mémorisées.

9. Run : Cette fonction exécute un programme mémorisé.

10. Edit, FWD, Back : Ces trois fonctions ont les mêmes rôles que dans le cas du robot Rhino XR-2.

11. Arm step : Cette fonction mémorise l'ampleur du mouvement relatif à un bras. Lorsqu'un mouvement est exécuté, il a une ampleur de ARM STEP.

12. Arm num. : Cette fonction mémorise le numéro de bras. Lorsqu'un mouvement est accompli, il est relatif au bras numéro ARM NUM. .

13. Translation step : Comme il a été spécifié ci-dessus, il est possible de mouvoir le robot selon trois axes. Lorsque le robot est déplacé selon un axe, quel qu'il soit, il l'est de TRANSLATION STEP unité (s).

Le lecteur peut se rendre compte de la similitude entre les commandes présentes sur le teach pendant du simulateur de robot et celles existantes généralement sur de véritables teach pendant.

Dès lors, l'utilisation de ce teach pendant ne devrait pas procurer de gros problèmes lorsqu'un programmeur de robot désirerait utiliser le simulateur pour tester la pertinence d'une suite de mouvements. En effet, le simulateur de robot se présente comme un bon outil de test. Avant d'appliquer une série de commandes sur de véritables robots, il pourrait être intéressant de vérifier leur validité. Cette opération pourrait se réaliser avec ce simulateur étant donné qu'il peut simuler tous les mouvements qu'un programmeur peut demander un vrai robot.

Une autre utilisation du simulateur, étant donné qu'il respecte quasiment les situations réelles, pourrait être l'apprentissage de la programmation d'un robot.

Il ne faut cependant pas oublier que dans la réalité, un robot fait partie d'un tout. Il est souvent en relation avec d'autres éléments qui peuvent être d'autres robots, un tapis roulant, ... Cet aspect n'est pas représenté par le simulateur; et c'est peut être la seule.

Chapitre 3

Concepts

CONCEPTS

Ce chapitre comprend une description du robot modélisé dans le logiciel de simulation et une présentation des caractéristiques de ce robot.

La première partie de ce chapitre est consacré à la description des différents référentiels qui permettent de localiser le robot tant dans l'espace réel que sur l'écran

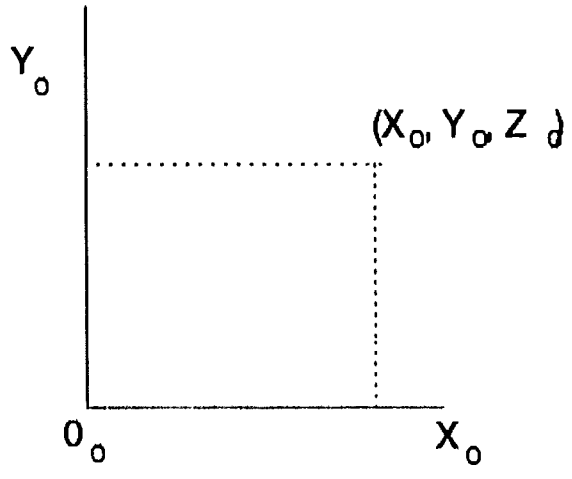
La deuxième partie contient la définition et un descriptif de tous les concepts significatifs dans la représentation du robot.

La troisième partie présente la représentation graphique du robot à l'écran.

3.1 SYSTEMES D'AXES

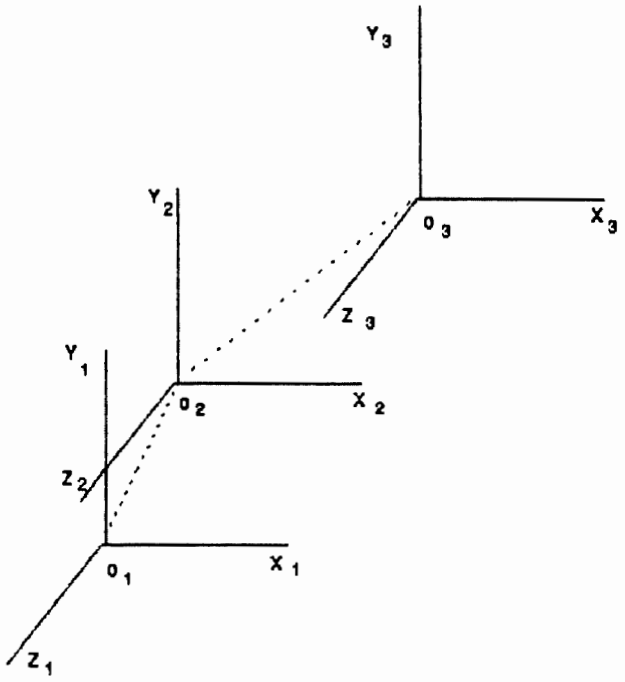
La représentation du robot à l'écran fait appel à 2 systèmes d'axes. Ces 2 systèmes d'axes interviennent pour la localisation du robot, d'une part sur l'écran, et d'autre part dans l'espace réel à l'intérieur duquel le robot se déplace.

Le premier système d'axes permet de déterminer l'emplacement du robot modélisé dans l'espace à 2 dimensions qu'est l'écran. Cependant, ce système d'axes se rapportant à l'écran est défini comme étant à trois dimensions et chaque point dans ce système d'axes est localisé par une coordonnée à 3 composantes, la troisième composante ayant un but particulier explicité par la suite.



Système d'axes du référentiel écran

Le deuxième système d'axes correspond en fait à un ensemble de systèmes de référence, chacun d'entre eux servant à localiser un bras du robot dans l'espace réel à 3 dimensions. Chacun des éléments de cet ensemble peut être considéré comme étant une translation du point origine et des 3 points directeurs le long des axes X,Y,Z du système du bras précedent.

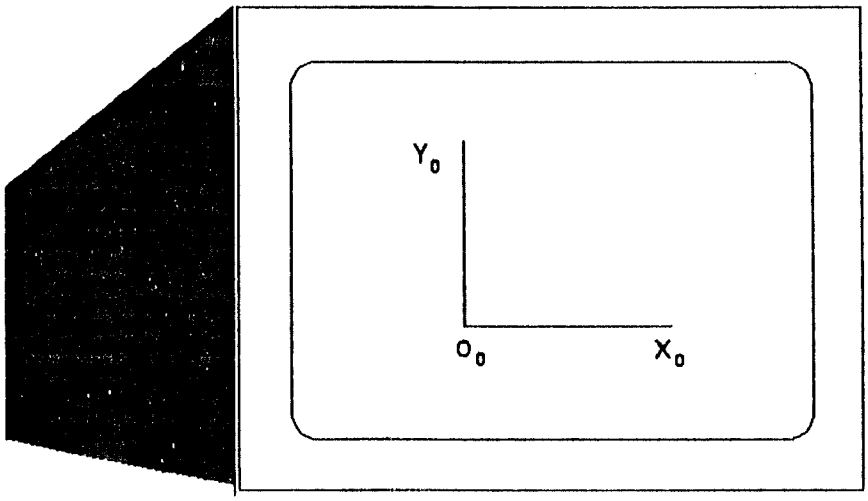


3.1.1 SYSTEME D'AXES DU REFERENTIEL-ECRAN

Ce système d'axes, appelé système d'axes du référentiel ecran ou du référentiel fixe correspond aux 3 axes (X_0,Y_0,Z_0), placés dans la position suivante à l'écran:

-l'axe X_0 est placé horizontalement sur l'écran.
-l'axe Y_0 est placé verticalemnt sur l'écran.
-l'axe Z_0 est une droite sortant perpandiculairement de l'écran.

On peut visualiser ce système d'axes de la manière suivante à l'écran:



L'axe Z_0 n'apparait pas sur la figure En eftet, puisque cet axe sort perpendiculairement à l'écran, sa représentation dans un plan à 2 dimensions correspond de ce fait à un point placé sur l'origine 0_0 des 3 axes.

Les 3 axes X_0, Y_0, Z_0 ont pour point d'origine le point 0_0, fixé à l'écran de façon arbitraire lors de la définition du robot.

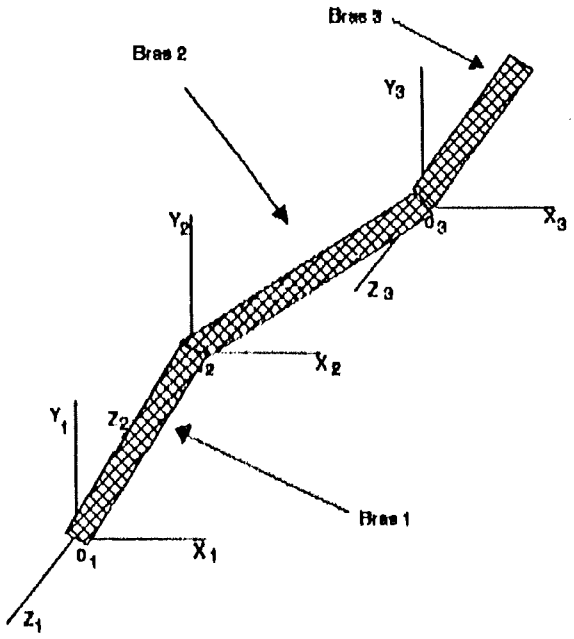
Ce ensemble doit servir à déterminer la position sur l'écran de tous les points nécessaires à la visualisation de chacun des bras du robot. Comme on l'a déjà signalé, cette position est exprimée par une coordonnée à 3 composantes dans le système d'axes.

Les composantes x_0 et y_0 de la coordonnée de chacun des points dans ce système d'axes donnent la position de ce point dans l'espace à deux dimensions constitué par l'écran. La composante z_0 de la coordonnée du point extrémité du bras doit servir à déterminer la grosseur du bras.

La notion de grosseur est prise en considération afin que l'observateur puisse se rendre compte à la seule visualisation du robot à l'écran, de la position relative de chacun des bras par rapport aux autres bras. Cette notion de grosseur permet de rendre la représentation du robot plus proche de la réalité par utilisation de la perspective.

3.1.2 SYSTEME D'AXES DU REFERENTIEL ACTIF

Pour pouvoir déterminer la position de chacun des bras dans l'espace réel à 3 dimensions dans lequel est placé le robot modélisé, on a recours à un ensemble de systèmes d'axes à 3 dimensions, soit un par bras .



Ensemble de systèmes d'axes.

3.1.2.1SYSTEME D'AXES DU ROBOT

Ce système d'axes est identique au système d'axes relatif au premier bras du robot.

Il est initialement placé sur celui du référentiel écran et les 3 axes X,Y,Z correspondent avec les axes X_0,Y_0,Z_0. Il a la particularité de pouvoir subir des rotations autour des 3 axes X_0,Y_0,Z_0 du référentiel écran. Ces rotations simulent les changements de point de vue pour l'observateur, témoin de la simulation.

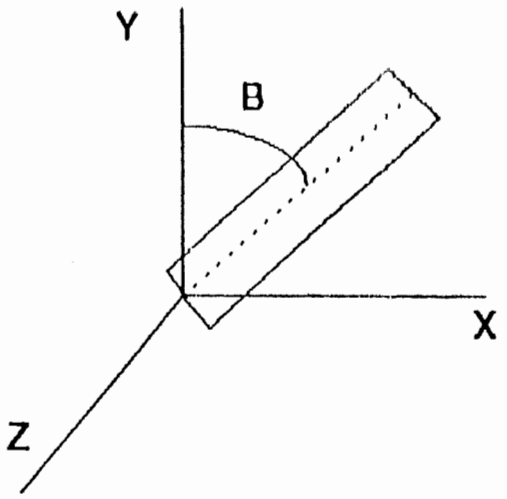
3.1.2.2SYSTEME D'AXES DES BRAS SUIVANTS

Pour chacun des bras suivants, on a également recours à un système d'axes permettant de déterminer la position de ce bras. Ce système (le système d'axes du bras suivant) a pour origine le point extrémité du bras précédent.

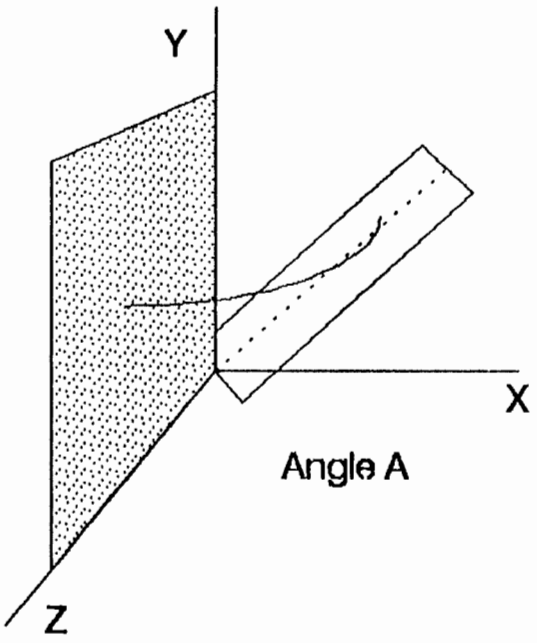
3.1.2.3 DEFINITION DE LA POSITION DU BRAS DANS SON SYSTEME D'AXES

La position du bras dans le système d'axes lui étant associé est déterminée à partir de la valeur des deux angles A et B.

L'angle A représente l'angle formé entre le plan déterminé par les axes Y et Z et le segment de droite représentant l'axe directeur du bras. Cette notion correspond à celle de méredien terrestre.



Angle entre bras et axe Y : angle B



Angle entre bras et le plan Y-Z : angle A

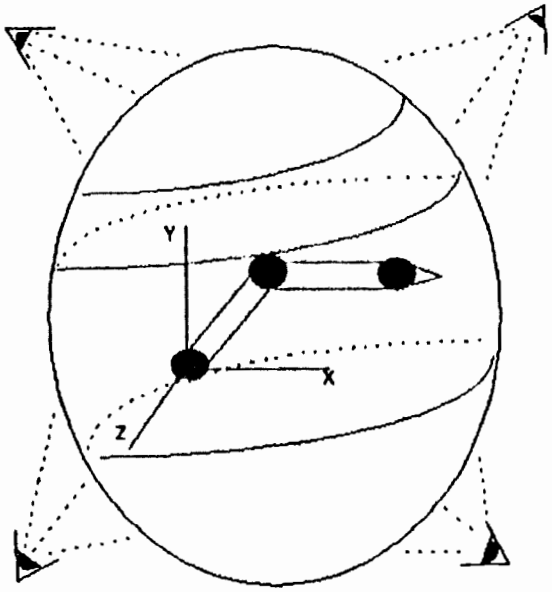
L'angle B représente l'angle formé entre l'axe Y et le segment de droite représentant l'axe directeur du bras. Cette notion correspond à celle de parallèle terrestre.

L'axe directeur du bras peut être défini à ce stade comme étant le segment de droite allant du point origine des axes jusqu'à l'extrémité du bras et la position du bras est définie par la position de cette extrémité (Cette notion sera approfondie lors de la description d'un bras au point 2.2).

3.1.3 CHANGEMENT DU POINT DE VUE

Une des particularités du simulateur est de permettre à l'observateur extérieur de pouvoir modifier l'angle de perception à partir duquel il voit le robot modélisé. On peut considérer que le robot est englobé dans une sphère dont le centre correspond au point d'attache du robot à la surface. L'observateur peut être placé en tout point de cette sphère et de ce fait voir le robot selon des angles de vue différents

Cette particularité est très importante si on veut modéliser une situation de la réalité où un observateur peut physiquement se déplacer autour du ro-



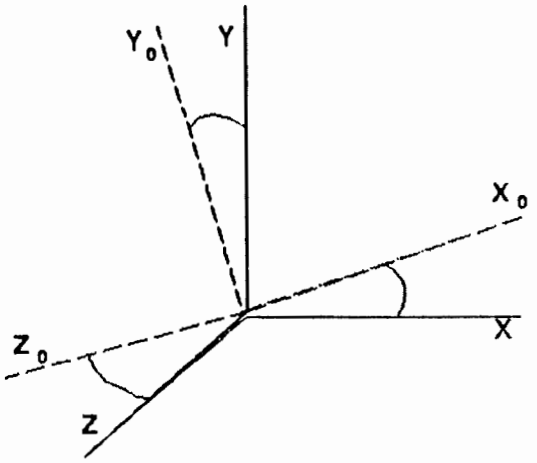
bot afin de le visualiser sous divers points de vue. De ce fait, l'observateur peut "tourner" autour du robot afin de mieux se rendre compte si la position atteinte correspond bien à celle désirée

Le changement de point de vue permet la visualisation de la position du robot selon différents points de vue. Cependant, il s'agit de visualiser la position

résultante et non l'exécution d'un mouvement à partir de différents angles de vision. L'observateur ne peut de cette manière effectuer de changements du point de vue que lorsque le mouvement en cours de simulation est terminé et que le robot a atteint la position résultante du dernier mouvement exécuté.

Le point de vue de l'observateur est déterminé par la position du système d'axes (X,Y,Z) du référentiel actif du premier bras par rapport au système d'axes (X_0,Y_0,Z_0) du référentiel écran et plus particulièrement des angles formés entre les axes X et X_0, Y et Y_0, Z et Z_0.

Un changement de point de vue peut être défini comme étant une suite de rotation des axes (X,Y,Z) autour des axes (X0,Y0,Z0). Ainsi, à tout mo-



ment, on peut dire que l'on fait subir une rotation d'un angle Xz autour de l'axe Z_0 du système d'axes (X,Y,Z), (0_x-360,0_y-360,0_z-360) pour déterminer la position du système d'axes (X,Y,Z) par rapport au système d'axes (X_0,Y_0,Z_0).

Tout changement de point de vue a pour effet de modifier la position des 3 points directeurs des axes (X,Y,Z) du référentiel actif du premier bras. Leurs coordonnées dans le système d'axes (X_0,Y_0,Z_0) forment la matrice de passage qui va permettre la transition du référentiel actif au référentiel écran. Toute modification du point de vue provoque de ce fait le calcul dune nouvelle matrice de passage d'un système d'axes à l'autre. Cette matrice de passage intervient dans le calcul de la position de chacun des bras à l'écran à partir de leur position dans l'espace réel.

Partant des coordonnées (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3) des 3 points directeurs du référentiel actif du premier bras. La matrice de passage est déterminée en calculant les nouvelles coordonnées de ces points dans le référentiel écran apres avoir fait une rotation d'un angle α_x (respectivement α_y, α_z) autour de l'axe X_0 (respectivement $Y_0 Z_0$) en appliquant les formules suivantes :

On pose (x_{i1},y_{i1},z_{i1}) la nouvelle coordonnée du point considéré et (x_i,y_i,z_i) l'anciènne coordonnée de ce point $(i=1..3, i_1=1..3)$.

Pour une rotation autour de l'axe X_0 d'un angle α_x, on obtient les nouvelles positions des 3 points directeurs en appliquant les formules suivantes à leurs coordonnées :

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_x & \sin\alpha_x \\ 0 & -\sin\alpha_x & \cos\alpha_x \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad i = 1..3,\ i1 = 1..3$$

ou

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = A_x \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \text{avec } A_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_x & \sin\alpha_x \\ 0 & -\sin\alpha_x & \cos\alpha_x \end{bmatrix}$$

Pour une rotation autour de l'axe Y_0 d'un angle α_y, on obtient les nouvelles positions des 3 points directeurs en appliquant les formules suivantes à leurs coordonnées :

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = \begin{bmatrix} \cos\alpha_y & 0 & \sin\alpha_y \\ 0 & 1 & 0 \\ -\sin\alpha_y & 0 & \cos\alpha_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad i = 1..3,\ i1 = 1..3$$

ou

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = A_y \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \text{avec } A_y = \begin{bmatrix} \cos\alpha_y & 0 & \sin\alpha_y \\ 0 & 1 & 0 \\ -\sin\alpha_y & 0 & \cos\alpha_y \end{bmatrix}$$

Pour une rotation autour de l'axe Z_0 d'un angle α_z, on obtient les nouvelles positions des 3 points directeurs en appliquant les formules suivantes à leurs coordonnées :

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = \begin{bmatrix} \cos\alpha_z & -\sin\alpha_z & 0 \\ \sin\alpha_z & \cos\alpha_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad i = 1..3,\ i1 = 1..3$$

ou

$$\begin{bmatrix} x_{i1} \\ y_{i1} \\ z_{i1} \end{bmatrix} = A_z \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad \text{avec } A_z = \begin{bmatrix} \cos\alpha_z & -\sin\alpha_z & 0 \\ \sin\alpha_z & \cos\alpha_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La nouvelle matrice de passage devient ainsi la matrice suivante:

$$A = A_x * A_y * A_z = \begin{bmatrix} x11 & x21 & x31 \\ y11 & y21 & y31 \\ z11 & z21 & z31 \end{bmatrix}$$

Le calcul de la position de chacun des bras à partir de cette matrice est présenté dans la partie "calcul des coordonnées" de ce chapitre.

Pour modifier la valeur des angles α_x, α_y, α_z l'observateur utilisera les touches du clavier telles que définies :

-les touches flèches haut et bas pour augmenter ou diminuer α_x

-les touches flèches gauche et droite pour augmenter ou diminuer α_y

-les touches Pgup et Pgdn pour augmenter ou diminuer α_z.

Les coordonnées des 3 points directeurs du système d'axes du référentiel actif du premier bras seront mémorisées dans un tableau.

Le changement de point de vue a pour effet de faire varier la représentation à l'écran du robot mais pas sa position dans l'espace réel. Le changement de point de vue correspond ainsi uniquement à une modification des coordonnées de chacun des bras dans le référentiel ecran. La position des bras dans leur référentiel actif n'est pas modifiée du fait que cette position est calculée partir des valeurs des angles A et B, et que ces valeurs ne sont pas modifiées par les rotations autour du système d'axes (X_0,Y_0,Z_0).

3.2 DEFINITION DU ROBOT.

3.2.1 REPRESENTATION DU ROBOT

Le robot dont le logiciel est chargé de simuler le comportement peut être considéré comme étant un ensemble de bras reliés entre eux par des organes de liaison. Le robot possède un seul point fixe et dispose à son extrémité d'un bras particulier qui joue le rôle d'organe actif.

Les bras peuvent être relié entre eux par deux types de liaison possibles:

-les rotules qui permettent au bras de se mouvoir dans toutes les directions autour du bras auquel il est relié.

-les charnières qui imposent au bras de ne se mouvoir que dans une seule direction autour du bras auquel il est relié.

La liaison avec la surface d'appui est également réalisée par l'intermédiaire d'une rotule ou d'une charnière. Un seul point fixe signifie que seul le premier bras du robot est en contact avec la surface d'appui. Cependant, on peut faire subir des translations à ce point fixe, ce qui a pour effet de déplacer en bloc tout le robot.

A l'extrémité de la suite de tous les bras, un bras particulier ayant des caractéristiques particulières est placé et ce bras joue le rôle de l'organe actif du robot réel simulé.

On appelle degré de liberté du robot ou axes de liberté, le nombre de directions indépendantes dans lesquelles le robot peut se mouvoir. Ces degrés de liberté correspondent aux directions dans lesquelles chaque élément constitutif du robot peut se déplacer ou s'orienter. Ces degrés de liberté sont déterminés par le type de liaison existant entre le bras concerné et celui auquel il est relié.

Le robot est défini à partir des éléments contenus dans deux tableaux définis logiquement.

Le premier tableau contient les informations nécessaires à la mémorisation des caractéristiques de chaque bras et au calcul de la position de ce bras dans son référentiel actif. Il contient de plus les positions de chacun des bras dans son référentiel actif et dans le référentiel écran.

Physiquement, on a réservé une zone mémoire dans laquelle on a sauvé le contenu des 2 tableaux logiques et à laquelle on accède directement pour y lire ou y mettre à jour les valeurs. Ce point est expliqué au chapitre ...

Le deuxième contiendra les informations nécessaires pour positionner l'origine de chacun des bras.

CONTENU DU PREMIER TABLEAU

Un bras est représenté par 3 éléments ou lignes de ce tableau logique. La signification de ces lignes est donnée dans la définition d'un bras.

Chaque ligne est constituée d'une suite de 17 nombres réels représentant les caractéristiques d'un point particulier d'un bras (On définit 3 points particuliers pour chaque bras).

Ces 17 nombres réels représentent:

- les composantes (x,y,z) de la coordonnée d'un point dans le référentiel actif du bras considéré. Ces composantes correspondent aux éléments 1, 2 et 3 du tableau.
- les composantes (x_0,y_0,z_0) de la coordonnée de ce point dans le référentiel écran et correspondant aux éléments 4, 5 et 6 du tableau.
- la valeur initiale de l'angle A pour ce point dans son référentiel actif. Cette valeur est contenue dans l'élément 7 du tableau.
- la valeur intermédiaire de cet angle A contenue dans l'élément 8 du tableau (cette valeur servira pour l'exécution d'un ordre particulier).
- la valeur courante de cet angle A contenue dans l'élément 9 du tableau (c'est à partir de cette valeur que l'on calculera la position du bras à tout moment).
- la valeur initiale de l'angle B pour ce point . Cette valeur est contenue dans l'élément 10 du tableau .
- la valeur intermédiaire de cet angle B contenue dans l'élément 11 du tableau (cette valeur servira pour l'exécution d'un ordre partculier).
- la valeur courante de cet angle B contenue dans l'élément 12 du tableau (c'est à partir de cette valeur que l'on calculera la position du bras à tout moment).

- la dimension du bras exprimé en unité et contenue dans l'élément 13 du tableau.
- une information indiquant si le bras est attaché à la surface d'appui (il s'agit de spécifier si l'origine du bras correspond au point fixe). Cette information est contenue dans l'élément 14 du tableau.
- le type de liaison entre ce bras et le bras précedent ou la surface d'appui s'il s'agit du premier bras. Cet élément correspond à l'élément 15 du tableau.
- les variations permises pour les angles A et B en fonction du type de liaison et des degrées de libertés fixés par l'observateur. Ce renseignement est contenu dans l'élément 16 du tableau.
- une indication des axes le long desquels des translations sont permises si ce bras est en contact avec la surface d'appui. Cette information est contenue dans l'élément 17 du tableau.

Ce tableau peut être aisément étendu afin de pouvoir y mémoriser des éléments supplémentaires tels que des bornes pour les angles A et B.

CONTENU DU DEUXIEME TABLEAU

Chaque élément de ce tableau logique correspond aux informations nécessaires pour positionner l'origine de chacun des bras.

Chaque ligne du tableau contient:

- les composantes (x,y,z) des coordonnées de l'origine du bras considéré dans le système d'axes du référentiel actif du bras précédent. Ces composantes sont mémorisées dans les éléments 1, 2 et 3 du tableau.
- les composantes (x_0,y_0,z_0) des coordonnées de l'origine du bras considéré dans le système d'axes du référentiel écran. Ces valeurs sont contenues dans les éléments 4, 5 et 6 du tableau.

On va maintenant déterminer de façon précise les concepts utilisés dans la définition d'un robot, à savoir:

- Le bras et sa représentation (point 3.2.2).
- L'organe actif (point 3.2.3).
- Le point fixe (point 3.2.4).

-Les rotule et charnière (point 3.2.5).

- Le dessin du bras à l'écran (point 3.2.6).

Le calcul des coordonnées des points nécessaires pour représenter le robot est présenté dans la partie 3 de ce chapitre. La présentation et la signification des mouvements du robot ou d'un bras seront présentées dans la partie 4 de ce chapitre.

3.2.2DEFINITION DU BRAS

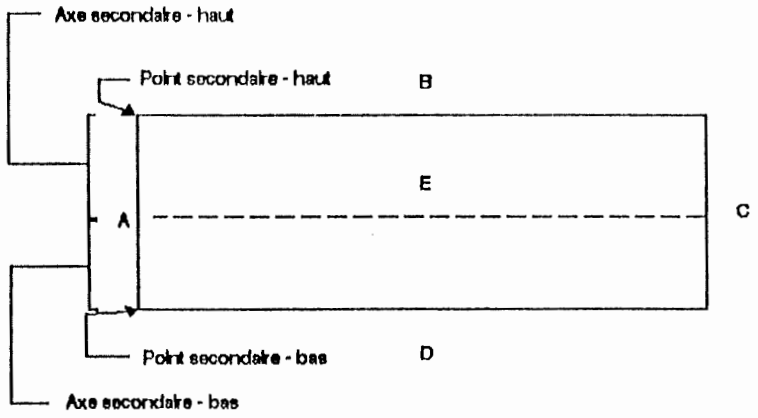
L'élément central du robot modélisé est le bras. Le robot est constitué d'une suite de bras. Chaque bras peut se mouvoir autour du point qui le relie au bras précédent dans certaines directions déterminées par un ensemble de contraintes.

Chaque bras est identifié par le numéro d'ordre correspondant à sa position dans la suite des bras. Le bras relié à la surface d'appui porte le numéro 1.

Pour des raisons de programmation, il est impossible d'avoir une infinité de bras. On a, de ce fait, limité le nombre de bras à un maximum de 5. Ce nombre est suffisant pour représenter la réalité vu que l'on trouve très rarement des robots constitués de plus de 5 bras. L'organe actif, placé à la suite des bras, est un bras distinct des autres bras et qui n'est pas repris dans les 5 bras maximum. Le robot modélisé ne peut avoir qu'un seul organe actif. Les limites imposées ont été choisies dans le but de simuler une majorité de robots possibles sans restreindre la généraliré.

Un bras est représenté à l'écran par le quadrilatère (a,b,c,d) ayant les caractéristiques suivantes :

- sa forme est la suivante



Représentation d'un bras de robot

-le côté a a une longueur en nombre de points déterminée par la grosseur du côté c du bras précédent s'il s'agit d'un bras autre que le premier. S'il s'agit du premier bras, ce côté a une longueur fixé en fonction de la carte graphique utilisée.

-les côtés b et d sont égaux et leur longueur est déterminée par la dimension du bras; un des deux côtés peut être considéré come étant la projection orthogonale selon la droite c de l'autre côté.

-le côté c a une longueur pouvant varier en fonction de la composante z_0 de la coordonnée du bras dans le référentiel ecran. Ce côté visualise la notion de grosseur.

Sur la figure de la page 3.12, un bras est représenté par un rectangle dont le côté c a la même longueur que le côté a mais lors de la simulation, il arrive souvent que la longueur du côté c soit différente de celle du côté a puisque cette longueur varie en fonction de la position du bras (notion de perspective). De ce fait, les angles entre les côtés a et b et b et c d'une part et d et c et d et c d'autre part ne sont pas nécessairement égaux à 90. L'organe actif est représenté par un triangle.

Les 3 éléments représentant un bras dans le tableau logique mémorisant le robot, définissent 3 points que l'on appelle respectivement point directeur, point secondaire-bas, point secondaire-haut. L'élément correspondant à un bras dans le tableau logique des origines détermine les coordonnées d'un point que l'on appèlle point origine. Le point extrémité d'un bras est défini comme étant son point directeur.

On appelle axe directeur ou axe principal, le segment de droite joignant le point origine au point directeur. On appèlle respectivement axe secondaire-bas et axe secondaire-haut, les segments de droites joignant respectivement le point origine au point secondaire-bas et au point secondaire-haut.

L'axe directeur est l'axe déterminant la position, l'orientation et la dimension du bras. Cet axe n'apparaît pas à l'écran mais est utilisé pour déterminer la position des 2 axes secondaires et pour le dessin du bras. Cet axe correspond à la droite c sur la figure page 3.12.

L'axe secondaire-bas et l'axe secondaire-haut sont 2 axes perpandiculaires à l'axe principal et dont la jonction correspond au segment de droite a sur la figure page 3.12.

La position du point directeur et des 2 points secondaires dans l'espace réel est déterminée par la valeur courante des angles A et B associées à ces points et contenues dans le tableau du robot.

Pour le point directeur, la valeur des angles A et B est fixée par l'utilisateur lors de la définition du robot modélisé et peut être modifiée par certains mouvements que l'on fait exécuter au bras en question. Pour le point secondaire-bas, l'angle A a la même valeur que celui du point directeur tandis que l'angle B a la valeur de celui du point directeur plus 90. Pour le point secondaire-haut, l'angle A a également la même valeur que celui du point directeur tandis que l'angle B a la valeur de celui du point directeur moins 90. L'actualisation des valeurs des angles pour les points secondaires est faite automatiquement à partir de l'évolution des valeurs de ces angles pour le point directeur.

Les mouvements du bras sont déterminés en fonction des contraintes contenues dans la ligne définissant le point directeur. Ces contraintes dépendent du type de liaison du bras

-avec la surface d'appui s'il s'agit du premier bras.
-avec le bras précédent s'il s'agit d'un bras autre que le premier.

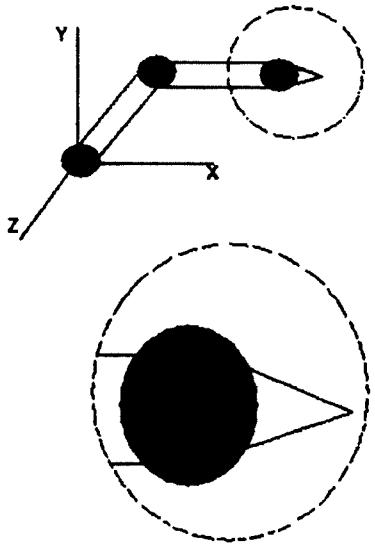
Les contraintes se rapportant aux mouvements des 2 points secondaires sont identiques à celles du point directeur. Ces contraintes n'ont aucune influence sur le comportement du bras considéré, seules les contraintes relatives au point directeur du bras étant prises en considération. Par souci d'uniformité et de complétude, ces contraintes sont toutefois enregistrées dans le tableau du robot.

Dans la suite de l'exposé, on désignera indifféremment par bras, une composante du robot correspondant à 3 lignes du tableau du robot et le quadrilatère représentant ce bras.

3.2.3 ORGANE ACTIF

L'organe actif peut être défini comme étant le bras placé à l'extrémité du dernier des bras constituant le robot. Le robot modélisé a un seul organe actif. Cependant, il n'est pas impossible d'imaginer un robot ayant plusieurs bras jouant le rôle d'organe actif. Ce type de robot serait nettement plus complexe, notamment en ce qui concerne l'identification du bras. Des idées d'adaptation aux robots ayant plusieurs organes actifs sont présentées à la fin de cet écrit.

Ce bras est défini comme un bras quelconque mais ayant les caractéristiques particulières suivantes :



Organe actif.

- la valeur des angles A et B est fixée à 90.
- la dimension de ce bras particulier est fixée arbitrairement.
- la liaison avec le bras précédent est obligatoirement du type rotule.
- les contraintes portant sur les angles A et B permettent à l'organe actif de se mouvoir dans toutes les directions.

La représentation à l'écran de l'organe actif se fait sous la forme d'un triangle.

L'objectif d'une telle représentation est de permettre de localiser les points par lesquels passe l'extrémité du robot lorsqu'on lui fait exécuter des mouvements. Une telle représentation de l'organe actif ne permet pas de visualiser la saisie d'un objet quelconque puisqu'il n'est pas possible de déterminer si l'organe actif est ouvert ou fermé.

D'une part, la visualisation de la saisie d'un objet demanderait des analyses et interprétations plus larges : il serait en effet nécessaire de se demander si

l'objet peut être saisi, si la forme de l'objet, sa masse, la résistance, ... permettent cette saisie, ...

D'autres part, il nous semble que la visualisation des points par lesquels passent l'extrémité du robot suffit pour évaluer une simulation puisque l'objectif principal est de connaître la trajectoire suivie et de la comparer avec celle souhaitée.

Afin de représenter la suite des points de passage et de pouvoir faire une analyse de ces points, les coordonnées de ces points sont mémorisées dans un fichier. On a ainsi la possibilité de faire une interprétation de ces points de passage. De plus, afin de permettre une meilleure visualisation, on a la possibilité de faire la liaison de tous ces points de passage par des segments de droite. On a, de ce fait, une visualisation de la trajectoire suivie par l'extrémité du robot. Ce fichier contenant la trajectoire est sauvée en même temps que le robot, son identifiant étant le même que celui du robot.

Les points de passage de l'organe actif peuvent également être affichés à l'écran ou sur papier sous une forme numérique.

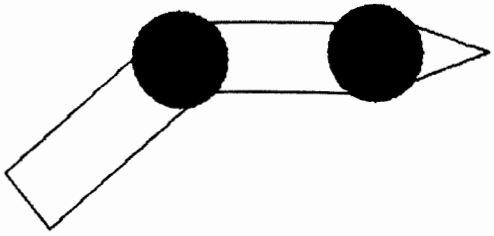
Lors de l'exécution de mouvement, l'organe actif est assimilé à un bras quelconque dont le numéro est celui du bras le précédent plus 1. On peut lui faire exécuter tous les mouvements permis pour un bras quelconque.

3.2.4 POINT FIXE ET SURFACE D'APPUI

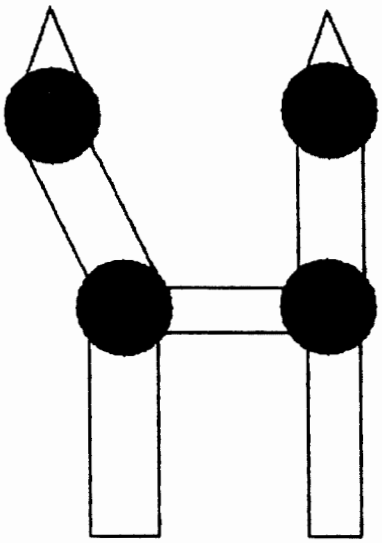
Le robot dont le comportement est simulé est un robot ayant un seul point en contact avec la surface sur laquelle il est posé. On appèlle point fixe ce point et surface d'appui la surface. Les robots à plusieurs points fixes demandent une gestion beaucoup plus complexe. Dans un premier temps, on s'est donc limité aux robots ayant un seul point fixe.

Pour modéliser des robots ayant 2 ou plusieurs points fixes, il faudrait adapter la représentation du robot ainsi que trouver une autre façon de localiser chaque bras. Des propositions d'adaptation sont faites au chapitre 7.

Le robot à un seul point fixe peut subir des mouvements sur la surface d'appui. On permet ainsi au robot de subir des translations le long des axes X, Y, Z du système d'axes du référentiel actif du premier bras.



Un point fixe



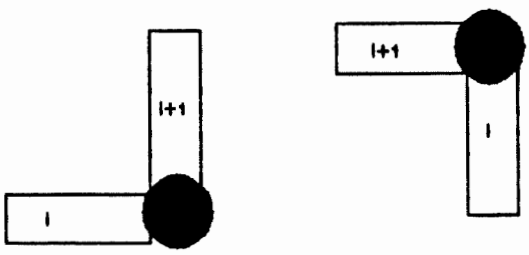
Deux points fixes

3.2.5 Rotule et charnière

Entre deux bras quelconques du robot, il doit exister un lien qui permette à chaque bras d'effectuer des mouvements indépendemment des autres bras tout en restant attaché au bras qui le précéde ou à la surface d'appui s'il s'agit du premier bras. Lorsqu'un bras change de position, les bras suivants ont deux posibilités de se comporter :

1- ils maintiennent la même position relative avec le bras ayant subi un mouvement:

L'angle formé entre les 2 bras reste constant lors du mouvement du bras i.



Même position relative

Cela signifie que les bras suivants changent de position et donc que les valeurs des angles A et B pour ces bras varient au cours du processus. Le programme calcule aussitôt la nouvelle position des bras mais ces formules ne font pas intervenir les angles A et B qui restent constants.

2- ils ne maintiènent pas la même position relative

Cela signifie que les valeurs des angles A et B des bras suivants ne

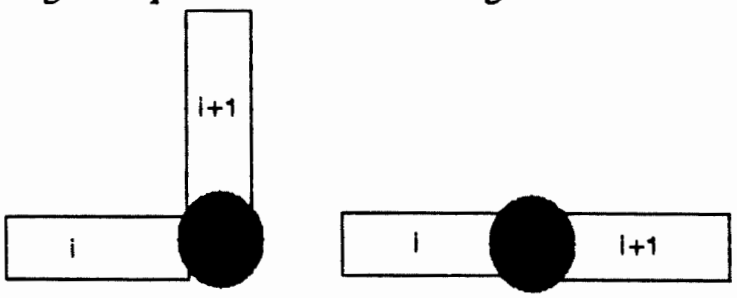


Fig 3.9 Changement de la position relative

sont pas modifiés. Ces bras changent de position du fait que leur origine aura été déplacée (l'origine du bras i+1 correspond à l'extrémité du bras i) mais l'angle entre les 2 bras a varié.

La première alternative correpond au comportement de la majorité des robots, l'autre alternative étant liée à certaines classes de robots particuliers. Cependant, dans un but de complétude, le logiciel donne la possibilité à l'utilisateur de choisir le comportement souhaité pour son robot.

Les mouvements du bras du robot dépendent essentiellement du type de liaison de ce bras avec le bras précédent, auquel il faut ajouter les contraintes portant sur la grandeur de ces mouvements exprimée en degrés.

3.2.5.1-LESROTULES

On peut définir une rotule comme étant une articulation de forme sphérique. Le bras fixé à une rotule peut ainsi, du fait de la sphéricité, se déplacer dans toutes les directions. Cependant, en raison de la manière choisie pour déterminer la position d'un bras en fonction des 2 angles A et B, les mouvements possibles sont définis en terme de variation des angles A et B.

Lorsqu'un bras est relié au bras précédent a l'aide d'une rotule, les mouvements de ce bras sont déterminés en fonction des contraintes suivantes :

- Variation de A uniquement : Cela signifie que la position du bras ne sera modifiée que par des variations de la valeur de l'angle A associé à ce bras.

- Variation de B uniquement : Cela signifie que la position du bras ne sera modifiée que par des variations de la valeur de l'angle B associé à ce bras.

- Variation de A et B : Cela signifie que la position du bras pourra être modifiée à partir de variations des angles A et B.

3.2.5.2-CHARNIERE

On peut définir une charnière comme étant un ensemble de 2 pièces métalliques assemblées sur un axe commun, l'une au moins des pièces étant mobile autour de l'axe.

Cela signifie qu'un bras relié au bras précédent par une charnière n'a la possibilité de faire des mouvements que dans un seul plan.

Ces mouvements sont interprété, du fait du choix posé pour déterminer la position d'un bras à partir de la valeur des angles A et B de ce bras, comme étant soumis aux contraintes suivantes :

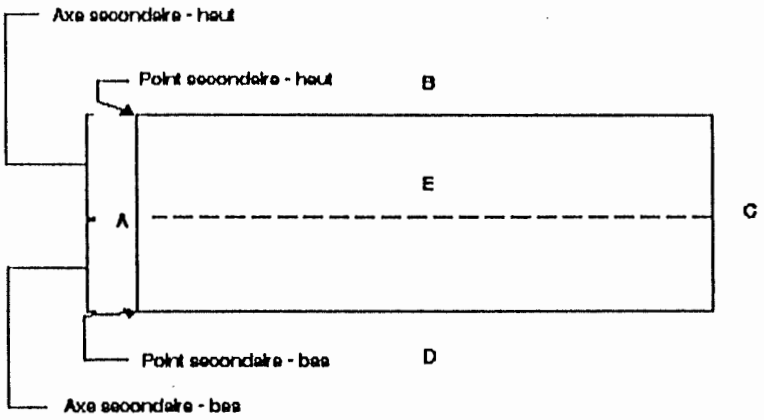
- variation de l'angle A uniquement.

- variation de l'angle B uniquement.

Quelque soit le type de liaison choisi, en plus des contraintes liées aux mouvements permis, il est toujours possible d'imposer des bornes aux variations des angles A et B. Par défaut, les angles A et B peuvent prendre toutes les valeurs comprises entre 0 et 360 mais il est possible que, pour un bras particulier, on impose que l'angle A et/ou l'angle B ne prennent des valeurs se situant dans un intervalle [m,n] avec $0 \leq m \leq n \leq 360$. Pour ce faire, il suffit d'étendre le tableau du robot et de prévoir des éléments pour chaque borne des angles.

3.2.6DESSIN D'UN BRAS

Pour dessiner un bras à l'écran, on utilise les coordonnées dans le référentiel écran des 3 points définissant un bras pour construire le quadrilatère visualisant le bras.



Représentation d'un bras de robot

On dessine le bras en traçant des segments de droite entre 4 points particuliers déterminés à partir des 3 points caractérisant un bras et dont les caractéristiques sont mémorisées dans le tableau du robot.

Deux de ces points sont les 2 points secondaires du bras. Le coté a du quadrilatère est formé de la jonction des 2 segments de droite joignant le point origine du bras avec les 2 points secondaires. La position de ces points sur l'écran est définie par les composantes x et y de leur coordonnée dans le référentiel écran

Les 2 autres points sont calculés à partir des informations connues au sujet du point extrémité et respectivement du point secondaire haut ou bas selon qu'il s'agit du segment b ou du segment d du quadrilatère. On fait également intervenir la notion de grosseur du bras pour déterminer la position de ces points.

La composante z de la coordonnée du point extrémité du bras dans le référentiel écran détermine la position du bras relativement aux autres bras vis à vis de l'observateur. Cette composante sert donc à déterminer la grosseur. Le nombre maximum de bras a été limité à 5 dans le but de permettre une représentation de la grosseur de manière significative.

La grosseur permet de déterminer le nombre de points dont il faut tenir compte pour déterminer la position des 2 points nécessaires pour tracer les segments b et d.

Le segment c est tracé en reliant les 2 points calculés par un segment dedroite.

3.3 CALCUL DES COORDONNES

3.3.1 CALCUL DES COORDONNEES DANS LE REFERENTIEL ACTIF D'UN BRAS

Comme décrit précédemment, un système d'axes a été défini pour chacun des bras. Les coordonnées d'un bras sont, de ce fait, déterminées dans le tableau logique contenant le robot comme étant calculées par rapport au système d'axes associé au bras traité.

Les coordonnées dans le référentiel actif donnent la position d'un point particulier dans l'espace réel à 3 dimensions correspondant à l'espace dans lequel le robot est placé.

Le calcul de ces coordonnées pour le point directeur et les 2 points secondaires se fait à partir des données suivantes :

- la dimension du segment de droite joignant le point considéré et le point origine du bras.
- la valeur courante de l'angle A relative à ce point.
- la valeur courante de l'angle B relative à ce point.

La dimension du segment de droite relatif au point directeur est une information fournie par l'utilisateur tandis que celle des 2 segments de droite relatifs aux 2 points secondaires est fixée arbitrairement.

Les valeurs des angles A et B correspondant à chacun des points directeurs des différents bras sont des informations fournies par l'utilisateur. Ces mêmes valeurs pour les 2 axes secondaires d'un bras sont déterminées à partir de celles du point directeur comme il a été spécifié précédemment.

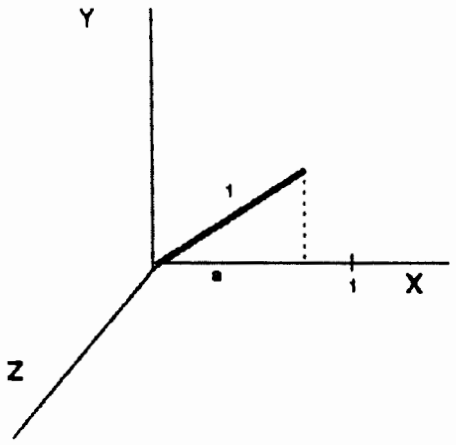
Le calcul des coordonnées des 3 points représentant un bras dans le référentiel actif de ce bras se fait de manière similaire. Nous ne décrirons le procédé que pour le point extrémité.

La valeur des coordonnées est calculée en appliquant la formule de trigonométrie : $a = l * \cos x$
où

l = longueur du segment de droite considéré,

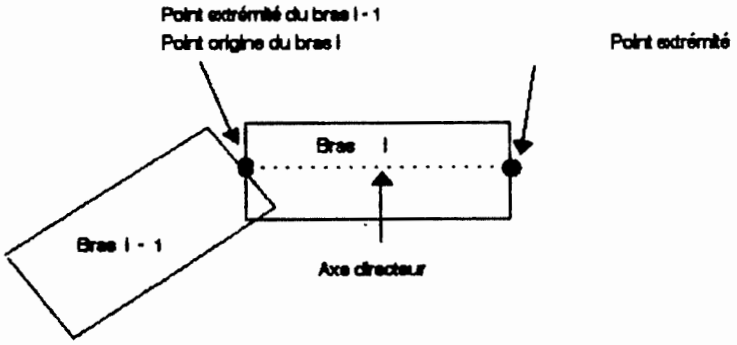
x = l'angle que ce segment forme avec l'axe sur lequel on désire avoir la valeur de la projection,

a = représente alors la longueur de la projection du segment sur l'axe par rapport au point unitaire de cet axe.



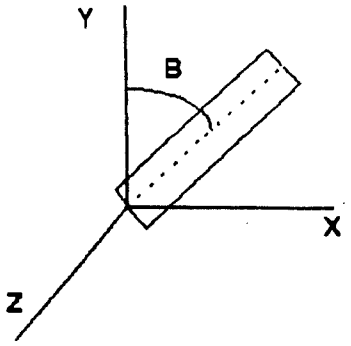
Calcul des coordonnées : méthode

Pour déterminer les coordonnées du point directeur et des 2 points secondaires, on considère le segment de droite reliant le point origine et le point considéré tandis que la dimension de ce segment est la dimension associée au point considéré dans le tableau du robot. Le calcul des coordonnées du point origine a été fait lors des calculs réalisés pour le bras précédent, l'origine du bras i correspondant à l'extrémité du bras i-1.



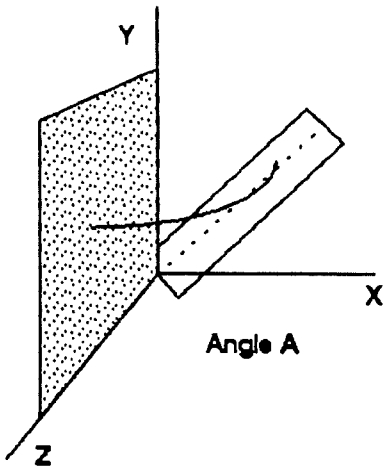
Calcul des coordonnées extrémités du bras I

La valeur de l'angle entre l'axe Y et le plan formé par les axes X et Z vaut 90. La valeur de l'angle entre l'axe Y et le segment de droite considéré vaut B par définition de l'angle B.



Angle entre bras et axe Y : angle B

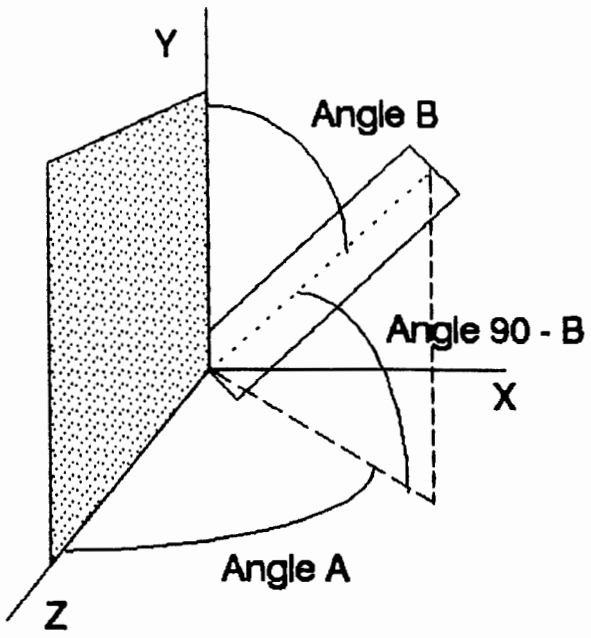
La valeur de l'angle entre le plan formé par les axes Y et Z et celui formé par les axes X et Z vaut 90. La valeur de l'angle entre le segment de droite considéré et le plan formé par les axes Y et Z vaut A par définition de l'angle A.

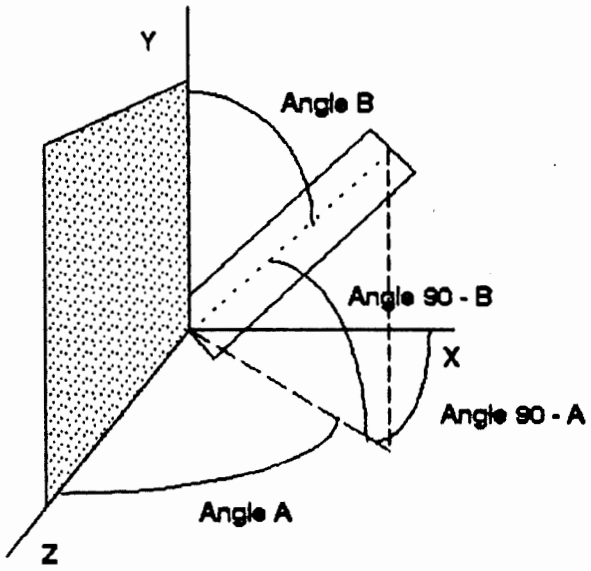


Angle entre bras et le plan Y-Z : angle A

On peut ainsi déduire que la valeur de l'angle formé entre le segment de droite considéré et sa projection sur le plan formé par les axes X et Z vaut 90 - B. Mais la valeur de l'angle formé entre la projection et le plan formé par les axes Y et Z n'est pas modifiée du fait de la projection et reste égale à A.

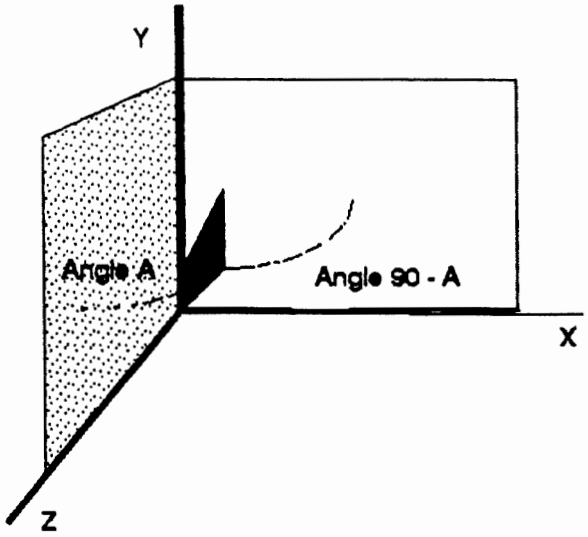
On peut également déduire que la valeur de l'angle formé par le segment de droite considéré et le plan déterminé par les axes X et Y vaut 90 - A. Dès lors, l'angle formé entre l'axe et sa projection dans le plan vaut 90 - A. D'autre part, la valeur de l'angle formé entre la projection du segment dans le plan XY et l'axe X vaut 90 - B puisque l'angle formé entre l'axes Y et l'axe X vaut 90 et l'angle formé entre l'axe Y et la projection du segment considéré dans le plan XY vaut B.





Les composantes (x,y,z) de la coordonnée d'un point peuvent être considérées comme étant la dimension de la projection du segment de droite associé à ce point sur les axes X, Y, Z respectivement.

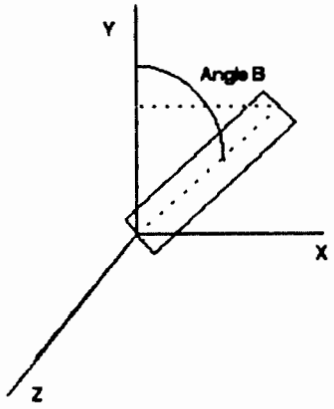
La composante x de la coordonnée du point considéré est calculée selon le raisonnement suivant:



La longueur de la projection du segment de droite dans le plan formé par les axes X et Y vaut DIM*COS(90-A).
La longueur de la projection de cette projection du segment dans le plan formé par les axes X et Y sur l'axe X vaut DIM*COS(90-A)*COS(90-B).

On peut donc déduire que la composante x de la coordonnée du point considéré dans le référentiel actif vaut
DIM*COS(90-A)*COS(90-B).

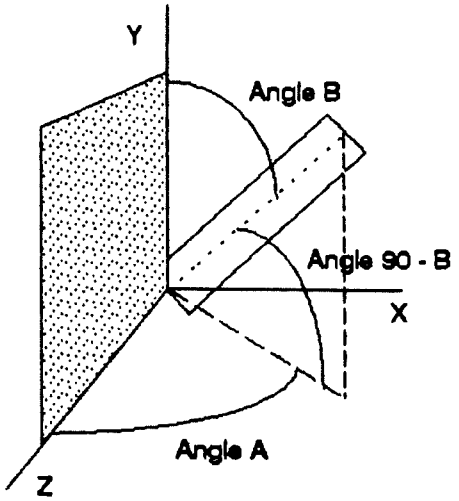
La composante y de la coordonnée du point considéré est calculée selon le raisonnement suivant:



La longueur de la projection du segment de droite sur l'axe Y vaut DIM*COS(B).

Il est possible de déduire que la composante y de la coordonnée du point considéré dans le référentiel actif vaut DIM * COS(B).

La composante z de la coordonnée du point considéré est calculée selon le raisonnement suivant:



La longueur de la projection du segment de droite dans le plan formé par les axes X et Z vaut DIM * COS(90 - b).
La longueur de la projection de cette projection de l'axe dans le plan formé par les axes X et Z sur l'axe Z vaut DIM*COS(A)*COS(90-B).

On peut donc déduire que la composante x de la coordonnée du point considéré dans le référentiel actif vaut
DIM*COS(A) * COS(90-B).

Après avoir appliqué ces formules aux 3 points déterminant un bras du robot, on peut compléter le tableau logique représentant le robot en y introduisant les composantes (x,y,z) de la coordonnnée des 3 points dans le référentiel actif de ce bras.

On répète ce procédé pour chacun des bras pour obtenir la position de chacun des bras dans leur référentiel actif. Cette position permet de déduire la position de chacun des bras dans l'espace réel.

3.3.2 CALCUL DES COORDONNEES DANS LE REFERENTIEL ECRAN

Les coordonnées dans le référentiel écran des 3 points définissant chaque bras sont nécessaires afin de permettre le dessin sur l'écran de ces bras. De plus, il est possible de représenter la notion de proximité du bras par rapport à l'observateur en utilisant la composante z_0 de la coordonnée du point directeur comme l'élément déterminant cette notion. Cette composante n'était pas nécessaire pour déterminer la position d'un point puisque l'écran étant un espace à 2 dimensions, une coordonnée à 2 composantes est suffisante pour placer le point sur l'écran. Cette coordonnée a cependant été ajoutée dans de but de visualiser de manière plus réaliste le robot puisqu'elle permet la représentation de la notion de perspective.

Pour calculer ces coordonnées, on va partir des coordonnées dans le référentiel actif de chacun des bras. Les coordonnées dans le référentiel actif d'un bras étant exprimées dans un système d'axes centré sur l'origine de ce bras, il est nécessaire d'exprimer ces coordonnées dans le référentiel actif du robot. Ensuite on pourra faire le passage du référentiel actif au référentiel écran en utilisant la matrice de passage définie ci-avant.

3.3.2.1 CALCUL DANS LE REFERENTIEL DU ROBOT

L'origine de chaque bras a été mémorisée afin de faciliter le calcul des coordonnées dans le référentiel actif du robot de tous les bras. L'élément i de ce tableau contient les coordonnées de l'origine du bras i exprimées dans le système d'axes du référentiel du robot. On obtient les coordonnées du point origine du bras i en faisant la somme des coordonnées dans le référentiel actif du robot, de l'origine du robot (coordonnées du point fixe) avec les coordonnées de chacun des points directeurs de tous les bras précédent le bras i considéré, dans leur référentiel actif.

Pour obtenir les coordonnées des 3 points définissant le bras i dans le référentiel actif du premier bras, il suffit d'ajouter les coordonnées de l'origine du bras considéré à celles de chacun des points dans le référentiel actif du bras i. On a ainsi les coordonnées du bras i exprimées dans le système d'axes du référentiel actif du premier bras.

3.3.2.2 PASSAGE DU REFERENTIEL ACTIF AU REFERENTIEL ECRAN

On a vu que le changement de point de vue correspondait à une rotation autour des 3 axes X_0, Y_0, Z_0 du système d'axes du référentiel écran.
Initialement, le système d'axes du référentiel actif du robot est situé sur celui du référentiel écran.

On exprime la position de chacun des points directeurs des 3 axes en fonction du système du référentiel écran et les coordonnées de ces 3 points constituent la matrice de passage d'un système d'axes à l'autre.

Si le point de vue n'a pas été modifié, cette matrice est la matrice unité à 3 lignes et 3 colonnes.
Par contre, lorsqu'on a fait varier le point de vue, cette matrice est calculée de la manière suivante :

Soient α_x, α_y, α_z les angles de rotations autour des axes X_0, Y_0, Z_0.

A_x, A_y, A_z les matrices correspondant aux rotations autour des axes X_0, Y_0, Z_0. (On a montré dans l'explication du changement de point de vue comment ces matrices étaient calculées).

x_i, y_i, z_i les composantes des coordonnées des points unités des 3 axes du référentiel actif du robot ($i = 1..3$).

Alors les coordonnées de ces 3 points directeurs dans le référentiel écran sont obtenues par le calcul suivant :

$$\begin{bmatrix} x_{i0} \\ y_{i0} \\ z_{i0} \end{bmatrix} = A_x * A_y * A_z * \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \qquad i = 1..3$$

et la matrice de passage est consituée par

$$\begin{bmatrix} x_{10} & x_{20} & x_{30} \\ y_{10} & y_{20} & y_{30} \\ z_{10} & z_{20} & z_{30} \end{bmatrix} = A$$

Où (x_{10},y_{10},z_{10}) (respectivement (x_{20},y_{20},z_{20}), (x_{30},y_{30},z_{30})) est la coordonnée du point unité de l'axe X (respectivement l'axe Y, l'axe Z) du système d'axes du référentiel actif du premier bras exprimée dans le système d'axes (X_0,Y_0,Z_0) du référentiel écran.

Pour obtenir les coordonnées de chacun des 3 points définissant un bras, il reste à multiplier cette matrice avec les coordonnées de ces points dans le référentiel actif du premier bras. On peut alors compléter le tableau du robot après avoir répéter le processus pour chacun des bras.

3. 4Description des mouvements

Un robot est une structure destinée à exécuter des mouvements dans l'espace à partir de commandes qui lui sont passées

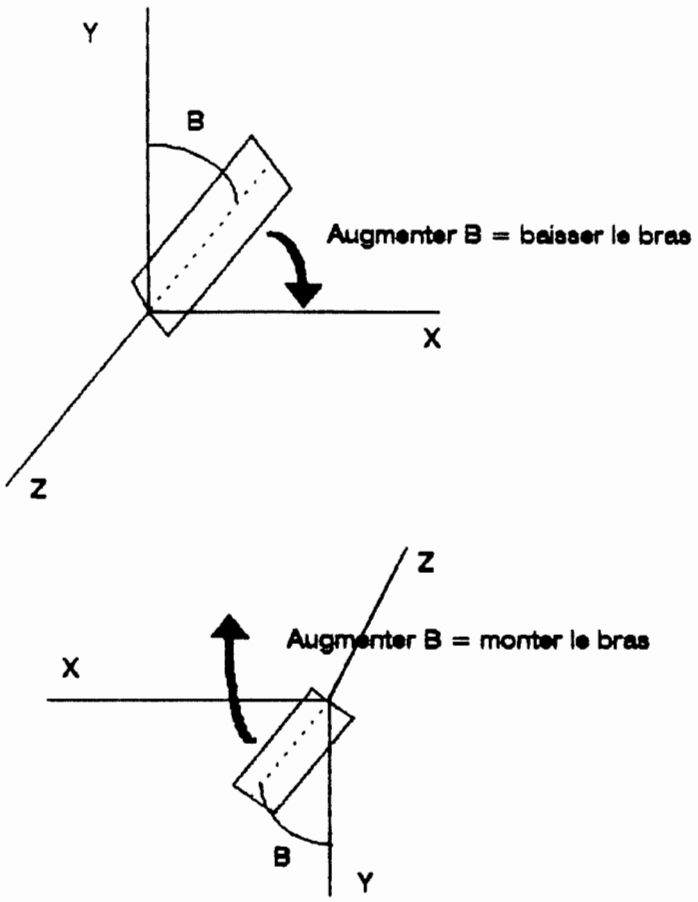
Dans cette partie, il est d_fini un ensemble de mouvements que le robot modèlisé sera amené à réaliser

3. 4.1Rotation d'un bras

Le premier mouvement que l'on peut simuler est le déplacement de l'extrémité d'un bras vers le haut ou vers le bas, vers la droite ou vers la gauche. Ce mouvement est appelé ROTATION

La rotation d'un bras est défini comme étant une variation quelconque de la valeur de l'angle A ou de l'angle B qui définissent la position du bras considéré. Toute rotation d'un bras corespond toujours à la modification d'un de ces 2 angles étant donné que la position d'un bras (plus précisément la position des 3 points déterminant un bras) est déterminé en fonction de la valeur de ces angles.

Une variation de A ou de B ne correspond pas nécessairement, en ce qui concerne la visualisation, à un déplacement du bras vers le haut ou vers le bas,



vers la gauche ou la droite. En effet, le point de vue peut avoir été modifié de telle manière qu'il n'est plus possible d'interpréter le mouvement du bras comme un mouvement selon les 4 points cardinaux. Cela justifie le choix fait pour déterminer la position de chaque bras à partir de la valeur des angles A et B.

Ce moyen de positionner un bras permet de déterminer la position de tout point indépendemment de l'endroit à partir duquel le robot est visualisé(point de vue de l'observateur).

Les influences possible de ce mouvement sur la valeur des angles A et B des bras suivant a été présenté dans le point 2 de ce chapitre.

3. 4.2 Pivotage d'un bras

Un autre mouvement important pour un bras de robot est la rotation sur lui-même. On appelle pivotage tout mouvement d'un bras du robot ayant pour effet de faire tourner sur lui-même ce bras et de faire tourner autour de ce bras les autres bras, le bras subissant le mouvement servant d'axes de pivotage.

Ce mouvement n'a aucun effet ni sur la position du bras concerné à l'écran ni sur sa visualisation du fait que l'on peut considérer que dans la réalité, un bras est un cylindre et que la représentation d'un cylindre dans un plan à 2 dimensions correspond à un quadrilatère qui reste identique lorsqu'il subit une rotation sur lui-même.

Par contre, les bras placés à la suite du bras subissant le pivotage, voient leur position dans la réalité et sur l'écran modifiées par le pivotage.

Pour calculer les nouvelles positions de chacun des points définissant les bras, une démarche différente de celle suivie pour le calcul des positions relatif aux autres mouvements est adoptée dans ce cas uniquement.

Pour calculer les nouvelles positions des bras succédant au bras ayant subi un pivotage, l'ancienne position du point directeur du bras qui a pivoté et la valeur de l'angle de pivotage du bras sur lui-même servent de point de départ pour le calcul de la matrice décrite ci-après. Par application de cette matrice à chacun des points directeurs des bras suivants, il est possible de calculer les nouvelles positions de ces points directeurs. Ensuite, de cette position, les nouvelles valeurs des angles A et B correspondant à cette position en sont déduites et les valeurs de ces angles pour les points secondaires sont mises à jour en fonction des règles résultant de la définition de ces points. La position de ces points secondaires est alors recalculée à partir des nouvelles valeurs des angles A et B.

Les formules de calcul de la nouvelle position d'un point à partir de son ancienne position et de l'angle de pivotage sont définies de la manière suivante :

Si · l'angle de pivotage = α

On pose $c = \cos\alpha$

$s = \sin\alpha$

$k = 1 - \cos\alpha = 1 - c$

Avec (e_x, e_y, e_z) = ancienne position du point directeur du bras subissant le pivotage

on pose $v_1 = e_x / \sqrt{(e_x^2 + e_y^2 + e_z^2)}$

$$v_2 = e_y / \sqrt{(e_x^2 + e_y^2 + e_z^2)}$$

$$v_3 = e_z / \sqrt{(e_x^2 + e_y^2 + e_z^2)}$$

On définit alors la matrice suivante

$$Rot = K * \begin{bmatrix} V1^2 & V1V2 & V1V3 \\ V1V2 & V2^2 & V3V2 \\ V1V3 & V2V3 & V3^2 \end{bmatrix} + \begin{bmatrix} C & -V3*C & V2*C \\ V3*C & C & -V1*C \\ -V2C & V1*C & C \end{bmatrix}$$

Si (p_x, p_y, p_z) est l'ancienne position, la nouvelle position (p_{x1}, p_{y1}, p_{z1}) est obtenue par application de la formule suivante :

$$(p_{x1}, p_{y1}, p_{z1}) = (p_x, p_y, p_z) * Rot$$

Les formules de calcul de la valeur des angles A et B relative à un point à partir de la coordonnée de ce point sont définies de la manière suivante :

Si (e_x, e_y, e_z) est la position courante d'un point, on d_finit les angles A et B à partir des formules suivantes:

on définit $A = Arccos(e_z / \_(e_z^2 + e_x^2))$

$$A = Arcsin(e_x / \sqrt{(e_z^2 + e_x^2)})$$

$$B = Arccos(e_y / \sqrt{(e_x^2 + e_y^2 + e_z^2)})$$

$$B = Arcsin(e_y / \sqrt{(e_x^2 + e_y^2 + e_z^2)})$$

3.. 4.3Translation du robot

Comme nous l'avons déjà dit, le robot modèlisé possède un seul point en contact avec la surface d'appui sur laquelle il est placé. Ce point est appelé point fixe.

Cependant, il est possible pour ce point fixe de se déplacer dans les 3 directions déterminées par le système d'axes du référentiel actif du robot. Ce mouvement est appelé translation du point fixe.

Ce mouvement consiste à déplacer le point fixe et donc tout le robot, le long d'un des 3 axes X, Y, Z du système d'axes du référentiel actif du robot. Il correspond en fait à une modification de la composante x, y ou z de la coordonnée dans le référentiel actif du robot, de l'origine du premier bras.

On pourrait faire subir des translations à chacun des bras au lieu du robot mais ce type de translation devrait être interprétée différemment : il s'agirait dans ce cas d'un allongement ou raccourcissement de la dimension du bras (ex. : on simule le comportement d'un verin hydraulique). Dans l'état actuel du programme, seules des translations du robot sont permises.

3.5 Positions particulières

Il est possible de définir des positions particulières dans lesquelle le robot peut être placé à un moment particulier de la simulation. Ces positions sont définies afin de permettre d'y revenir à tout instant.

3.5.1 Position initiale

La position initiale est mémorisée lors de la création du robot par le sauvetage des valeurs de départ des angles A et B relatifs à chacun des bras constitutifs du robot. Cette position est définie par rapport à un référentiel actif qui est superposé sur le référentiel écran.

3.5.2 Position intermédiaire

A tout moment, un utilisateur peut trouver intéressant de retenir la dernière position dans laquelle se trouve le robot modélisé et ce dans le but de lui faire reprendre cette position ultérieurement. La position sauvegardée est appelée position intermédiaire du robot.

Tant qu'il n'y a pas eu de sauvetage d'une position intermédiaire, celle ci a les valeurs de la position initiale.

Lors du sauvetage de cette position, on ne cherche pas à connaître la position du système d'axes (X,Y,Z) du référentiel actif du robot par rapport à celle du système d'axes (X_0,Y_0,Z_0) du référentiel écran. Cela n'est pas nécessaire car lors

du retour à cette position, le robot est replacé dans le système d'axes courant du référentiel du robot.

Chapitre 4

Programme résident

4.PROGRAMME RESIDENT

4.1 Un operating system multi-tâche

MS-dos fut conçu pour exécuter un programme à la fois. La possibilité d'exécuter plusieurs programmmes simultanément (operating system multi-tâche) n'était pas prévue. Un operating system uni-tâche est plus facile à concevoir qu'un operating system multi-tâche. De plus, le micro-processeur 8088 était trop lent et ne pouvait adresser assez de mémoire pour supporter efficacement le multi-tâche. Les utilisateurs de PC devaient donc se résigner à exécuter un seul programme à la fois.

D'un autre côté, DOS supporte des routines qui permettent aux programmes lors de leur terminaison, de rester résidents en mémoire. C'est ce que l'on appelle des programmes résidents. Leur utilisation est vue comme étant une forme de multi-tâche. Ces routines permettent à un programme de rester présent en mémoire dans un état passif (DOS ne le reconnait pas comme étant celui qui est en de s'exécuter) tandis qu'un autre programme en mémoire est actif, c'est - à - dire, DOS le reconnait comme étant celui qui s'exécute. Ces services étaient préalablement prévus pour supporter des drivers.

Malgré l'existence de ces services de TSR (Terminate/Stay Resident), ce n'est que plus tard que les programmes résidents connurent un certain développement. PRINT.COM fut l'un des premiers programmes résident. Il permet à l'utilisateur d'imprimer un fichier en arrière fond, en même temps qu'il utilise un autre programme. C'est une forme de multi-tâche.

Depuis lors, le marché des programmes résidents a continué de grandir. Sidekick, le champion indiscuté des programmes résidents, définit le standard pour ces types de programmes, et par la suite, d'autres suivirent mais avec une qualité pas toujours égalée. A l'heure actuelle, la majorité des PC ont différents programmes résidents "en exécution".

4.2 Le concept de programme résident

En général, lorsqu'un programme termine son exécution, la mémoire qu'il utilisait est libérée et l'utilisateur voit apparaître le prompt du DOS. Cette règle ne vaut pas pour les programmes résidents.

Les programmes résidents passent par deux états : un état actif et un état passif. Ils passent d'abord dans un état actif et se termine par un appel à l'un

des 2 services, 27 H ou 31 H, qui permettent à un programme de rester résident en mémoire dans un état passif.

Le service 27 H restreint la taille du programme à 64 K (Kilo bytes = 1024 bytes). Le service 31 H, par contre, permet des programmes plus gros et permet aussi au programme de retourner un code de sortie (exit code) au DOS. Quelque soit le service choisi, il faut dans les 2 cas calculer la quantité de mémoire dont le programme a besoin. Heureusement, l'unit DOS du Turbo Pascal comprend une procédure nommée Keep qui réalise ce calcul et appelle automatiquement le service 31 H du DOS.

4.3 Que se passe-t-il après un "Keep" ?

La fonction Keep bloque simplement le programme en mémoire et ce, là où il se trouve et dans un état passif.

Le programme résident peut faire à nouveau quelque chose d'utile, mais il doit être activé.

Il existe en général 3 manières d'activer un programme résident. La première est par l'horloge interne du système, qui bat 18.2 fois par seconde. Le programme est activé à chaque "battement" de l'horloge. C'est la technique qui est employée pour le programme PRINT-Com, mais pas avec une aussi haute fréquence d'activation.

La seconde manière d'activer un programme résident est par le clavier. Il suffit de presser une combinaison de touches, appelée les "hot key", et le programme résident est activé, ce qui se traduit souvent par l'apparition d'un écran du programme. C'est la technique employée pour des logiciels tels que Sidekick et autres.

Enfin, le troisième et dernière manière d'activer un programme résident est de court circuité les interruptions réservées par DOS pour l'utilisateur, celles-ci allant de 60 H à 67 H. Il suffit de faire pointer une de ces interruptions vers le programme résident de sorte que l'exécution de cette interruption correspond à l'exécution du programme résident. Cette dernière technique est celle utiliséedans la réalisation de notre logiciel.

4.4Permutation des registres

Un point délicat pour les programmes résidents est la commutation des registres. Un programme utilise des registres pour accéder aux instructions suivantes ainsi qu'aux données présentes en mémoire. Si ces registres sont confus,

indéterminés, le programme va rechercher l'information dont il a besoin à une mauvaise place.

Quand un programme résident est activé, le stack segment (SS) et le stack pointer (SP) (2 registres) contiennent les valeurs relatives au programme interrompu. Si le programme résident s'exécute avec le(s) stack(s) d'un autre programme, cela ne peut se solder que par des problèmes.

Pour les surmonter, le programme résident doit changer les valeurs dans SS et SP par ses propes valeurs et mémoriser les valeurs des registres du programmes interrompus. Quand le programme résident est sur le point de se terminer, il doit remettre les valeurs du programme interrompu. Il en a la possibilité étant donné qu'elles ont été préalablement sauvées.

4.5 Garder traces.

La stratégie de commutation des stacks qui vient d'être décrite juste ci-avant demande la mémorisation des valeurs de SS et SP des programmes résidents. Sans cette étape de mémorisation, il n'est pas possible, le moment venu, d'initialiser les registres à leurs valeurs adéquates.

Il est aussi utile de sauver une autre information avant de "locker" le programme dit résident, à savoir les valeurs originales de chaque vecteur d'interruption que le programme résident a capturé..

Il y a 2 raisons pour lesquelles il faut mémoriser ces valeurs. Premièrement, quand une interruption est capturée pour l'usage d'un programme résident, l'opportunité de faire appel à l'interruption originale doit toujours être offerte, sans quoi un résultat défavorable risquerait de se produire. Deuxièmement, si le programme résident, à un certain moment, est enlevé de la mémoire, les valeurs originales du vecteur doivent être restaurées.

4.6 Activation et communication avec le programme résident: explication.

Le système DOS a réservé des vecteurs d'interruption pour l'utilité du programmeur; ces vecteurs allant de 60 H à 67 H. Un de ces vecteurs permet d'activer et de passer de l'information au programme résident, à savoir le simulateur de robot. Quand le programme résident est, pour la première fois, chargé en mémoire, il capture un de ces vecteurs d'interruption qui est libre et le fait pointer vers une routine de communication dans le programme résident. Cette routine a la

capacité d'activer le programme résident; c'est en quelque sorte un centre nerveu, ou encore un dispatcher. Elle permet d'exécuter des procédures et fonctions en fonction de l'information reçue.

Avant de réserver une interruption comprise entre 60 H et 67 H, il est nécessaire de déterminer quelle est celle qui va être utilisée. En réserver une au hasard n'est pas très pratique étant donné qu'elle peut déjà être réservée par un autre programme résident.

Seule une interruption qui n'est pas utilisée, peut être capturée. Elle pointera vers la routine de communication du programme résident.

Une interruption qui peut être utilisée par un quelconque programme a sa valeur égale à nil. Pour trouver un vecteur non utilisé, il suffit de passer en revue les vecteurs d'interruption en commençant par l'interruption 60 H. Quand un vecteur dont la valeur est égale à nil est trouvé, il est certain qu'aucun autre programme résident n'est en train de l'utiliser. Cette interruption est capturée et pointe vers la routine de communication du programme résident. En même temps, le numéro de l'interruption capturée est sauvé.

Par exemple, l'interruption 63 H est libre, le programme résident doit se souvenir que c'était l'interruption 63 H qu'il avait prise, de sorte qu'il puisse la libérer une fois le programme terminé.

Une fois que le programme résident est chargé, l'interruption capturée est utilisée pour communiquer des instructions au programme résident.

Comment un programme actif, quelqu'il soit, peut-il passer de l'information au programme résident ? Cela se fait par le registre AX. Le programme résident a une routine de communication liée à l'interruption 63 H, par exemple. En assignant une valeur dans la partie AH du registre AX et en appelant cette interruption, il est possible de communiquer la fonction que vous voulez que le programme résident exécute. Sur base de ce registre, la routine de communication activera la fonction qui correspond à la valeur du registre AX.

Exemple :

```
@RET1CM80C = Procedure communication;
        case AH of
        0: fonction1;
        1: fonction2;
        .....
        end;
```

Cette technique est celle utilisée réaliser la communication entre le programme de l'utilisateur et le simulateur de robot. Pour de plus amples détails, le lecteur voudra bien se référer au chapitre "Communication entre deux programmes".

4.7 Connaissance du numéro d'interruption par le programme de l'utilisateur

Une fois le vecteur d'interruption choisi par le programme résident, il doit être communiqué au programme interrompant, c'est à dire le programme créé par l'utilisateur.

Cette opération se réalise par l'intermédiaire d'un fichier connu par les 2 programmes.

Pour de plus amples renseignements sur la communication du vecteur d'interruption, le lecteur peut se référer au chapitre intitulé **communication entre les 2 programmes.**

4.8 Plusieurs erreurs critiques au niveau du DOS peuvent stopper le programme résident :

4

4.8.1 Erreurs critiques

Ces erreurs surviennent quand DOS rencontre un événement "perturbateur" qu'il ne peut réellement traiter.

Un accès à un disquette se trouvant dans un drive où la porte n'est pas fermée est la source d'une erreur critique. DOS vous répond :

```
Not ready error reading drive A
About, Retry, Ignore ?
```

Si vous êtes à l'intérieur d'un programme et que vous tapez A pour Abort, votre programme résident (will bomb) "explosera" à moins que vous ayez court-circuité l'interruption n 24 h liée aux erreurs critiques. Si l'erreur critique se passe alors que le programme résident est actif, cela fermera (will lock up) votre système. Vous vous devez de vous protéger contre de tels problèmes.

Alors que ces problèmes sont sérieux, la manière pour les résoudre est simple. Dans votre programme résident, vous devez déclarer une

procédure d'interruption pour remplacer l'interruption n 24 h. Cette procédure fera juste une chose : mettre le registre AX à 0. En effet, faire cela revient à dire au DOS qu'aucune erreur ne s'est produite. Etant donné ce court-circuit, vous devez être capable d'assurer que le programme ne générera pas de telles erreurs.

4.8.2 Le "contral-break"

Quand l'utilisateur presse les touches CTRL et BREAK en même temps, DOS arrête le programme actif. Vous ne voulez pas que cela arrive dans votre programme résident. De nouveau, la solution est simple.

Le contrôle du Control Break peut être mis à "on" et "off" avec le service DOS 33 h. En mettant 1 dans AL et 0 dans DL et en appelant le service 33 h, le control break n'est plus possible.

Cela est très intéressant, mais le programme résident ne devrait pas changé le statut du control break pour un programme extérieur. Dès lors, si le programme "appelant" a rendu possible le control break, le programme résident doit le rendre de nouveau actif quand il se termine.

Cela est également assez simple. Avant que votre programme résident ne désactive le control break, il devrait déterminer le statut courant. Cela est aussi fait par le service DOS 33 h, mais cette fois, AL est mis à 0. Quand le service a terminé, le statut courant est contenu dans le registre DL. Sauver cette valeur de sorte que vous pouvez la restaurer quand le programme résident finit.

4.9 Déchargement du programme résident

A un certain moment, vous voulez décharger votre programme résident de la mémoire et libérer cette mémoire pour que d'autres programmes puissent l'utiliser.

Mais il risque de se poser quelques problèmes si d'autres programmes résidents sont présents en mémoire suivant le programme résident qui va être déchargé. Dès lors, il est nécessaire de déterminer si d'autres programmes résidents ont été chargés après le programme résident concerné par le déchargement.

Pour le vérifier, il faut contrôler les vecteurs d'interruption qui furent capturés lorsque votre programme résident a débuté. En général, tout programme prend l'interruption relative au clavier, relative au timer et à l'interruption n 28 h. Si un programme résident était chargé après le votre, il serait presque certain qu'il aurait aussi capturé une de ces interruptions. Mais attention, le simulateur (programme résident) ne capture aucune des trois interruptions citées ci-dessus. Il ne réserve qu'une interruption parmi les interruptions 60 H à 67 H. Dès lors, il est impossible de se rendre compte s'il existe d'autres programmes résidents suivant le simulateur, ce qui nous fait dire que le déchargement du simulateur libére également la mémoire de tout autre programme suivant le simulateur.

La première étape du déchargement du simulateur est de replacer l' interruption qui a été capturée lors du chargement dans son état initial. Il est aisé de restituer sa valeur vu qu'elle fut sauvée lorsque le programme débuta.

La seconde étape consiste à libérer la mémoire que DOS a alloué au programme résident. Cette libération se fait à l'aide de deux softwares : RELEASE et MARK. Le programme MARK a pour fonction de mettre une "marque" au début de la zone mémoire libre. Le programme résident est chargé au-delà de cette marque. Lorsqu'il s'agit de libérer la zone mémoire occupée par le programme résident, le programme RELEASE est exécuté, sa fonction étant de libérer toute la zone mémoire se situant au-delà de la marque.

Chapitre 5

Interface

5. INTERFACE PROGRAMME DE COMMANDES - INTERPRETEUR GRAPHIQUE.

L'objectif du programme de simulation étant de visualiser le comportement d'un robot suite à l'exécution d'un ensemble de commandes transmises par l'utilisateur, il est nécessaire de spécifier les différentes commandes que l'on peut lui faire exécuter. Il est également nécessaire de définir les informations à connaître pour les faire exécuter. Il est également utile de spécifier comment ces commandes sont fournies au programme chargé de faire l'interprétation graphique.

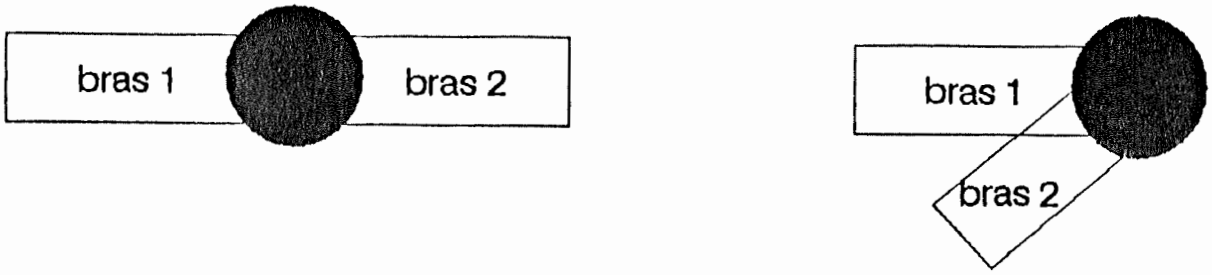
La première partie de ce chapitre consiste en une présentation des différentes commandes que l'utilisateur peut spécifier dans son programme de commandes. Ces commande sont définies d'une part pour faire exécuter les mouvements de tout ou partie du robot et d'autre part pour réaliser des ordres plus généraux servant pour l'analyse de la trajectoire. Cette description comprend le relevé de toutes les informations nécessaires à leur exécution correcte et la description des étapes logiques de leur exécution.

Une des particularités du système de simulation réside dans le fait que ces ordres sont communiqués à partir d'un programme auxiliaire developpé dans ce seul but et qui est indépendant du programme de simulation. On attire l'attention sur le fait que les ordres donnés au robot doivent être syntaxiquement corrects et qu'aucune validation syntaxique n'est prévue dans l'état actuel de développement du logiciel. Il faut donc que l'utilisateur veille à donner les ordres selon le format qui sera préciser dans la suite et à respecter les contraintes imposées. La communication entre l'interpréteur graphique et le programme de commande est présenté dans la deuxième partie de ce chapitre.

Les commandes doivent être placées à un endroit connu des 2 programmes. Le programme RESIDENT doit connaître cet endroit pour pouvoir aller les chercher et pouvoir les interpréter. Le programme auxiliaire qui a pour but de définir ces commandes et les constituer, doit donc les placer en un endroit d'accès connu et dont on est sûr qu'il ne sera pas utilisé à une autre tâche. La description de l'endroit où les commandes sont placées est également présentée dans la première partie de ce chapitre.

Dans l'état actuel du système, il n'est pas possible de faire une interprétation sémantique des commandes. De ce fait, toute commande syntaxiquement correcte pourra être visualisée sans vérifier si la position résultante est susceptible d'exister dans la réalité. Dans la suite, il sera possible d'ajouter un module qui aura pour effet d'interpréter une commande et de vérifier si elle est exécutable en fonction des contraintes qui peuvent exister sur un robot réel. En particulier, un module ayant pour objectif de vérifier si des bornes fixées pour les

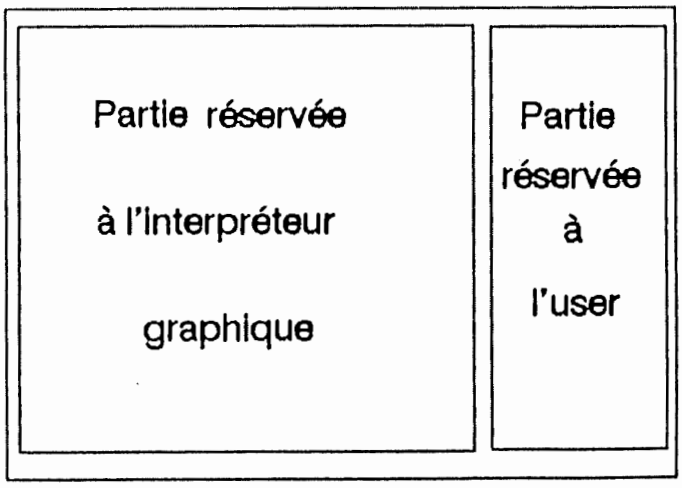
angles A et B sont respectées pourra être adjoint au programme. En effet, lors de la création d'un robot ou même en cours de simulation, des bornes inférieures et supérieures pourraient être fixées de manière définitive ou temporaire pour la valeur de chaque angle et il pourrait être nécessaire de vérifier si ces contraintes sont bienrespectées.



Ainsi, pour ce robot, le bras 2 pourrait ne pouvoir prendre des valeurs comprises entre 45 et 315 et ce à cause de l'objet de liaison qui impose des limitesmatérièlles.

Le programme RESIDENT peut être vu comme étant un interpréteur graphique chargé de visualiser les commandes passés à un robot à partir du programme de commande.

Ce programme RESIDENT visualise le robot sur la partie de l'écran qui lui est allouée. En effet, l'écran est partagé en 2 parties, l'une étant affectée au programme de simulation tandis que l'autre est réservée au programme de commandes.



Répartition de l'espace écran

5.1.DESCRIPTION DES COMMANDES

5.1.1 ROTATION D'UN BRAS

La première commande définie permet de réaliser le mouvement de rotation d'un bras tel qu'il a été défini dans le chapitre 3.

Pour faire exécuter ce mouvement à un bras du robot, il faut connaître les informations suivantes :

- le numéro du bras ; tous les bras peuvent subir de modifications de la valeur de leurs angles A et B, y compris l'organe actif dont le numéro lors de l'exécution des commandes est défini comme étant le numéro du dernier bras plus 1.

- l'angle que l'on fait varier : il s'agit soit de l'angle A, soit de l'angle B. Il n'est pas possible d'avoir un mouvement où on pourrait faire varier par une seule commande les 2 angles A et B. Cela n'est pas impossible à imaginer mais nous n'avons retenu que des variations d'un seul angle à la fois.

- le signe de la variation : + ou - .

- la amplitude de la variation : la variation ne peut dépasser 360. C'est en effet un non-sens que de vouloir faire exécuter des mouvements de plus de 360. Un angle de 360 correspond à un tour complet; Une variation de x avec x 360 est donc identique à une variation de x - 360 et l'on obtiendra un robot dans la même position que si on avait réalisé une variation de x.

L'exécution de ce mouvement a pour effet de provoquer les actions suivantes :

- calculer la nouvelle position dans le référentiel actif du bras qui subit le mouvement. Cela est nécessaire puisque cette position est calculée à partir des angles A et B et qu'un de ces 2 angles a subi une variation. Ce calcul es également nécessaire pour les bras suivants si le choix de

la première possibilité de variation des angles A et B a été fait lors de la configuration du programme (Voir Chapitre 3, point 2).

- calculer la nouvelle position dans le référentiel écran de ce bras. Lorsque la position dans le référentiel actif est modifiée, celle dans le référentiel écran est automatiquement modifiée. Il faut donc actualiser la position du bras dans le référentiel écran lors de chaque modification dans le référentiel actif.

- mettre à jour les coordonnées des points origines des bras suivants. L'origine du bras suivant immédiatement le bras considéré est modifiée puisqu'elle correspond à l'extrémité du bras. Celles des bras suivants sont modifiées par répercution de la modification de la position du bras.

- calculer les nouvelles positions dans le référentiel écran des bras suivants.

- visualiser le robot dans sa nouvelle position.

5.1.2 PIVOTAGE D'UN BRAS

La deuxième commande implémente le mouvement de pivotage d'un bras tel qu'il a été défini dans le chapitre 3.

Pour faire exécuter cette commande à un bras du robot, il faut connaître les informations suivantes :

- le numéro du bras qui subit une rotation sur lui-même. Chaque bras, y compris l'organe actif, peut subir cette rotation sur soi-même même si cela n'a pas beaucoup de sens dans le cas-ci. En effet, l'exécution de ce mouvement ne modifie la représentation que des bras suivants celui qui subit le mouvement. Comme dans ce cas, il n'y plus de bras à la suite, aucune modification n'apparaîtra lors de la visualisation.

- le signe du pivotage. Le signe '+' correspond à un pivotage du bras concerné dans le sens des aiguilles d'une montre tandis que le signe '-' correspond à un pivotage de ce bras dans le sens contraire.

· la valeur de l'angle de pivotage. Cette valeur doit également être inférieure à 360 et ce pour la même raison que pour la variation des angles A et B.

L'exécution de ce mouvement a pour effet de provoquer les actions suivantes :

· calculer la position des points directeurs du bras concerné et des bras suivants à partir des formules présentées ci dessus.

· déduire la valeur des angles A et B pour ces points à partir des formules données précédemment.

· mettre à jour les valeurs des angles A et B des points secondaires pour chaque bras selon les contraintes spécifiées lors de la définition des points secondaires haut et bas de chaque bras.

· calculer la position de ces points secondaires en fonction de la valeur des angles A et B selon la méthode suivie lors des autres mouvements.

· mettre à jour le tableau des origines de bras. Cela est nécessaire puisque les bras suivants le bras ayant subi le pivotage ont vu leur position modifiée.

· calculer la position des bras dans le référentiel écran.

· visualiser le robot dans sa nouvelle position.

5.1.3 TRANSLATION DU ROBOT

La troisième commande définie permet de réaliser le mouvement de translation du robot le long d'un des 3 axes X, Y ou Z du système d'axes du référentiel du robot.

Pour l'exécution de mouvements, il est nécessaire de connaître les informations suivantes :

· l'axe le long duquel se fait la translation. Lors de la création du robot ont été spécifiés les axes le long desquels une translation est possible. La gestion de cette contrainte sera faite dans le module d'interprétation des commandes qui sera ajouté ultérieurement.(Voir chapitre 7). Dans l'état actuel du système, cette contrainte doit être gérée par l'uti-

lisateur lui même puisque les commandes sont sensées être correctes sémantiquement.

- le signe de la translation.

- la grandeur . La translation totale le long d'un axe a été fixée à 100 points de chaque côté de l'origine des axes. Cette limite est arbitraire et relève d'un souci de restreindre la difficulté de gestion tout en conservant une échelle de translation pouvant simuler une situation réelle.

L'exécution de ce mouvement a pour effet de provoquer les actions suivantes:

- répercuter la translation dans la composante x, y ou z de la coordonnée de l'origine du premier bras dans son référentiel actif en fonction de l'axe choisi et de la grandeur de la translation.

- mettre à jour les origines des autres bras en repercutant la translation.

- calculer les nouvelles positions à l'écran de chacun des bras.

-visualiser le robot dans sa nouvelle position.

5.1.4SAUVETAGED'UNEPOSITIONINTERMEDIAIRE

La quatrième commande correspond au sauvetage de la position intermédiaire. La signification de cette position a été présentée dans le chapitre 3.

Le sauvetage de cette position se fait en mémorisant les valeurs des angles A et B dans les éléments prévus à cet effet dans le tableau du robot (respectivement les éléments 8 et 11) de chacune des lignes.

Pour ce mouvement, il n'est pas nécessaire de connaître des informations particulières puisqu'il s'agit uniquement de modifier en partie le tableau du robot et toutes les informations nécessaires sont connues.

5.1..5RETOURALA POSITION INITIALE

La cinquième commande se rapporte à un mouvement de retour à la position que le robot occupait lors de sa création.

Le retour à la position initiale a pour effet de calculer la position de chacun des bras à partir de la valeur des angles A et B données lors de la création du robot.

Ce mouvement ne nécessite aucune information particulière pour son exécution.

L'exécution de ce mouvement provoque les actions suivantes :

·affecter les valeurs initiales des angles A et B aux valeurs courantes de ces mêmes angles. Ces valeurs initiales sont mémorisées dans les éléments 7 et 10 de chacune des lignes du robot tandis que les valeurs courantes sont placées dans les éléments 9 et 11 du tableau du robot.

·calculer la nouvelle position dans le référentiel actif de chacun des bras du robot. Cela est nécessaire puisque les valeurs des angles A et B ont été modifiées.

·calculer les nouvelles coordonnées de chacun des points origines des bras dans le référentiel écran.

·visualiser le robot dans sa position initiale.

5.1.6 RETOUR A LA POSITION INTERMEDIAIRE

De la même manière qu'il a la possibilité de revenir à la position initiale du robot, l'utilisateur peut revenir à une position intermédiaire qu'il aura préalablement définie et mémorisée dans le tableau du robot.

Le retour à cette position intermédiaire a donc pour effet de placer chacun des bras en fonction des valeurs intermédiaires de leurs angles A et B. Ces valeurs intermédiaires ont été mémorisées dans le tableau du robot.

Les actions à exécuter pour réaliser cette commande sont similaires à celles du retour à la position initiale si ce n'est que les valeurs intermédiaires des angles A et B sont affectées aux valeurs courantes de ces mêmes angles et non les valeurs initiales.

5.1.7.AFFICHAGE DES COORDONNEES DE LA TRAJECTOIRE A L'ECRAN

A un certain moment dans la similation, l'utilisateur peut vouloir connaître les coordonnées de tous les points par lesquels est passée l'extrémité du

robot. Ces coordonnées ont été mémorisées au fur et à mesure de l'exécution des mouvements dans un fichier placé dans le disque virtuel.

Si le robot a été construit par chargement d'un robot déjà existant, il est possible que la trajectoire suivie lors de la dernière simulation soit mémorisée également. L'utilisateur a eu la possibilité, lors de la lecture du robot, de garder cette ancienne trajectoire. Le fichier de trajectoire contient donc toujours les valeurs correspondantes.

Cette commande a pour effet de lire le contenu du fichier de sauvegarde de la trajectoire et d'afficher l'enregistrement souhaité. L'utilisateur peut choisir le point désiré en utilisant les touches 'flèche haut' et 'flèche bas'.

Pour cette commande, il n'est pas nécessaire de connaître des informations particulières puisqu'elle n'a aucune influence sur la position du robot.

5.1.8. AFFICHAGE DES COORDONNEES DE LA TRAJECTOIRE SUR PAPIER

Pour faire l'analyse de la trajectoire suivie par l'organe actif, il peut être intéressant de disposer des coordonnées de chacun des points par lesquel est passée cette trajectoire sur papier.

Cette commande a pour effet de lire le fichier dans lequel est mémorisé la trajectoire et de l'imprimer sur papier selon un format défini au préalable.

L'exécution de cette commande ne demande la connaissance d'aucun paramètre particulier.

5.2 Communications entre programmes.

5.2.1 Introduction

Comme il a été écrit dans l'introduction de ce chapitre rappelant la spécification du programme, RESIDENT.PAS est un interpréteur graphique amélioré de commandes. Ces commandes sont toutes issues d'un programme créé par un utilisateur.

Ce dernier est supposé fournir des commandes à cet interpréteur mais en respectant un certain format et en assurant le contenu de la commande. Pour rappel, une commande passée à l'interpréteur graphique est composée de 4 informations :

- le type de mouvement.
- l'angle concerné par le mouvement, à savoir l'angle A ou l'angle B; ou l'axe (X,Y,Z) s'il s'agit d'une translation.
- le signe du mouvement (+ ou -).
- le numéro du bras concerné par le mouvement.
- la grandeur du mouvement. Cette valeur est comprise entre une borne inférieure et une borne supérieure. La valeur de ces bornes dépend du type de mouvement.

L'utilisateur peut ignorer totalement les détails relatifs à la communication avec l'interpréteur. Sa seule contrainte est de fournir des commandes dont la validité syntaxique et sémantique est assurée. Un interface destinée à la communication avec l'interpréteur graphique est à sa disposition.

Cet interface est une librairie de procédures et de fonctions. L'utilisateur se voit obligé d'utiliser ces différentes procédures et fonctions s'il désire transmettre ses commandes à l'interpréteur graphique. Passons en revue le contenu de cette librairie :

5.2.2 Communication : librairie de procédures et fonctions.

5.2.2.1 XXINITIALISATION (XXOUTIL)

Déclaration : procedure xx_initialisation.

Usage : xx_initialisation.

Paramètres : Néant

Fonction : La procédure xx_initialisation initialise le mode graphique nécessaire à l'exécution de l'interpréteur graphique (Hercules,Cga,Ega,Vga) et détermine le "fenêtrage" de l'écran.

Remarque : A la lecture du second objectif, le lecteur pourrait se poser la question de savoir ce que signifie le terme "fenêtrage". Pour mémoire, il faut se rappeler que l'écran de l'ordinateur est divisé en deux parties. L'une est réservée à l'interpréteur graphique, l'autre au programme de l'utilisateur.

Restrictions : Il est interdit à l'utilisateur d'insérer dans son programme une instruction destinée à initialiser la carte graphique. La fenêtre réservée à l'utilisateur est toujours de type texte. Il lui est dès lors impossible d'utiliser des procédures et fonctions graphiques dans son programme.

Voir aussi : XX_END.

Exemple :

```
program utilisat;
begin
xx_initialisation;
--
end.
```

5.2.2.2 XX_MADE_ROBOT (XXOUTIL)

Declaration : procedure xx_made_robot.

Usage : xx_made_robot.

Paramètre : Néant.

Fonction : Cette procédure est utilisée lorsque l'utilisateur se décide à travailler avec un robot.

Restriction : Cette procédure ne peut être invoquée que si la procédure XX_INITIALISATION a déjà été appelée. Dans le cas contraire, cela aurait pour conséquence de stopper l'interpréteur graphique.

Remarque : L'utilisateur fait appel à cette procédure au début de son programme lorsqu'il désire travailler avec un robot. Il lui est également possible d'appeller cette procédure au milieu de son programme s'il désire changer de robot.

Dans ce cas, si l'utilisateur désire, au préalable, sauver le robot courant, il doit appeller la procédure XX_SAVE qui s'en chargera (cfr supra).

Voir aussi : XX_SAVE.

Exemple

```
Program utilisat;
begin
        xx_initialisation;
        xx_made_robot
                ...
        write ('Nouveau robot (o/n) : ');
        read(ch);
        if ch = 'o'
        then begin
                write ('Sauver robot (O/N) : ');
                read(ch);
                if (ch = 'O') or (ch = 'o') then xx_save;
                xx_made_robot;
        end;
        ...
        xx_end;
end.
```

5.2.2.3. XX_END (XXOUTIL)

Declaration: procedure xx_end;

Usage: XX_END.

Paramètre: Néant.

Fonction: Cette procédure est utilisée pour fermer le mode graphique préalablement initialisé par la procédure XX_INITIALISATION. Elle décharge également la mémoire de l'interpréteur graphique.

Restriction: Lorsque XX_END est appelé, le programme resident se termine ainsi que le programme créé par l'utilisateur. Dès lors, XX_END sera la dernière instruction du programme de l'utilisateur.

5.2.2.4 XX_SEND_COMMAND (XXOUTIL)

Déclaration : procedure xx_send_command (mouv,angle,signe : char; numbras,valeur : integer).

Usage: xx_send_command (mouv,angle,signe,numbras,valeur).

Paramètres :

MOUV - ce paramètre indique le type de mouvement que l'interpréteur doit exécuter. MOUV peut prendre les valeurs suivantes :

R : rotation du bras n x d'un angle A (ou B) de + Y degré(s).

T : translation du point d'attache du robot selon l'axe X (ou Y ou Z) de x unités.

P : pivotage du bras n x de y degré(s)

S : sauvetage de la position courante comme position intermédiaire vers laquelle il est possible de revenir.

I : retour à la position intermédiaire préalablement sauvée.

H : retour à la position home, c'est - à - dire la position spécifiée lors de la construction du robot.

A : affichage à l'écran des coordonnées dans le référentiel actif et écran des coordonnées des points de la trajectoire suivi par l'organe actif du robot. Cette opération n'est possible que si la trajectoire est affichée à l'écran. Il faut, pour obtenir ce résultat, exécuter le programme de configuration et répondre "oui" lorsqu'il est demandé si la trajectoire doit être affichée ou non.

M : impression des coordonnées dans les référentiels actif et écran des coordonnées des points de la trajectoire suivie par l'organe actif du robot. Même remarque que pour l'affichage en ce qui concerne la trajectoire. C'est une précondition à l'impression.

ANGLE - Ce paramètre indique l'objet référentiel suivant lequel le mouvement sera exécuté. L'objet référentiel peut être l'angle A ou l'angle B, l'axe X ou l'axe Y ou l'axe Z. ANGLE n'a pas nécessairement une valeur. Sa valeur est fonction du type de mouvement transmis. Si le mouvement est :

- Rotation, alors ANGLE se voit attribuer comme valeur soit A ,soit B.
- Translation alors ANGLE se voit attribuer comme valeur soit X, soit Y, soit Z.
- Pivotage, sauvetage de la position intermédiaire, retour à la position intermédiaire (si elle existe), retour à la position home, affichage et impression des coordonnées, dans ce cas la valeur de ANGLE peut être indéterminée (l'utilisateur est libre de ne pas lui attribuer une valeur étant donné que l'interprétation ne tiendra pas compte de ce paramètre).

SIGNE - ce paramètre indique si la valeur du mouvement est positive ou négative. SIGNE prendra, par conséquent, la valeur '+' ou '-'. mais tout comme ANGLE, il ne prendra pas dans tous les cas une valeur. Lors d'un sauvetage de la position intermédiaire, d'un retour à la position intermédiaire, d'un retour à la position home, d'un affichage et d'une impression des coordonnées des points de la trajectoire suivi par l'organe actif, la valeur de SIGNE peut être indéterminée.

NUMBRAS - Ce paramètre spécifie le numéro du bras qui fait l'objet du mouvement. A nouveau pour ce paramètre, deux situations différentes coexistent.

1) Si le mouvement est une rotation ou un pivotage, NUMBRAS est le numéro du bras qui subit le mouvement. La valeur de NUMBRAS doit être comprise entre 0 et nombre de bras spécifié lors du choix ou de la construction du robot.

2) Pour les autres mouvements, il n'est pas utile d'assigner une valeur à ce paramètre. Soit la commande est relative à l'ensemble du robot (retour position... , sauvetage, ...), soit elle est relative uniquement à l'organe actif (affichage et impression des coordonnées des points de la trajectoire suivie par l'organe actif).

VALEUR - ce paramètre indique la grandeur du mouvement. Comme pour les autres paramètres exception faite de MOUV, sa valeur est fonction du type de mouvement.Si le mouvement est :

- Rotation et pivotage, VALEUR sera compris entre 0 et 359 (degré(s)).

-Translation, VALEUR sera compris entre 0 et 100 (unités).

-Sauvetage et retour position intermédiaire, retour position home, affichage et impression des coordonnées des points de la trajectoire suivie par l'organe actif, VALEUR peut ne pas être assigné d'une valeur. Il n'en sera pas tenu compte lors de l'interprétation.

Voyons un résumé des différentes valeurs qu'il faut assigner aux paramètres pour respecter la sémantique d'une commande

cfr tableau récapitulatif des commandes page suivante

Une présentation plus détaillée de ces commandes est faite dans la première partie de ce chapitre.

Fonction : XX_SEND_COMMAND assure la transmission d'une commande vers l'interpréteur graphique.

Restriction : Avant de transmettre une quelconque commande, il faut avoir fait appel à la procédure XX_INITIALISATION ainsi qu'à la procédure XX_MADE_ROBOT. Cela représente un non sens de vouloir communiquer une commande s'il n'existe aucun robot ou si le mode graphique n'est pas intialisé, sans quoi il est impossible de représenter un robot à l'écran.

Exemple : 1) Quelques formats d'appel de cette procédure :

Remarque : "?" valeur indéterminée

xx_send_command(T,x,+,?,20)

xx_send_command (R,a,-,1,10)

. . . . . .

Tableau récapitulatif des commandes

Paramètre / Commande	Mouvement	Angle	Signe	Nombres	Valeur
Rotation	R	A	+	1 ... 5	> 0 et < 360
	R	A	-	1 ... 5	> 0 et < 360
	R	B	+	1 ... 5	> 0 et < 360
	R	B	-	1 ... 5	> 0 et < 360
Translation	T	X	+	?	0 ... 100
	T	X	-	?	0 ... 100
	T	Y	+	?	0 ... 100
	T	Y	-	?	0 ... 100
	T	Z	+	?	0 ... 100
	T	Z	-	?	0 ... 100
Pivotage	P	?	+	1 ... 5	0 ... 360
	P	?	-	1 ... 5	0 ... 360
Position intermédiaire Sauvetage	S	?	?	?	?
Retour	I	?	?	?	?
Changement point de vue	0	?	?	?	0 ... 360
Retour position home	H	?	?	?	?
Affichage coordonnées	A	?	?	?	?
Impression coordonnées	M	?	?	?	?

2)Structure d'un programme:

```
Program utilisat;
begin
        xx_initialisation;
        xx_made_robot;
        tant que envoi commande est vrai
        begin
                choix de la commande;
                xx_send_command(mouv,angle,signe,numbras,va-
                leur);
        end;
        Si sauvetage robot est vrai
                then xx_save;
        xx_end;
end.
```

5.2.2.5. XX_GET_INFO (XXOUTIL)

Déclaration: function xx_get_info (code,numbras: integer) : real.

Usage: xx_get_info (code,numbras).

Paramètres :

NUMBRAS : cfr commande xx_send_command. Remarque importante : Si l'utilisateur décide d'avoir des informations sur l'organe actif, NUMBRAS doit être égal à 0.

CODE : ce paramètre indique le type de donnée désiré en retour lors de l'appel de cette fonction.

Fonction : La fonction XX_GET_INFO renvoie une donnée relative au bras du robot numero NUMBRAS en fonction d'un CODE. Voyons quelle est la donnée retournée en fonction d'un CODE déterminé :

1 = > coordonnée x du bras n NUMBRAS dans le référentiel actif.

2 = > coordonnée y du bras n NUMBRAS dans le référentiel actif.

3 = > coordonnée z du bras n NUMBRAS dans le référentiel actif.

4 = > valeur initiale de l'angle A du bras n NUMBRAS.

5 = > valeur intermédiaire de l'angle A du bras n NUMBRAS.

6 = > valeur courante de l'angle A du bras n NUMBRAS.

7 = > valeur initiale de l'angle B du bras n NUMBRAS.

8= >valeur intermédiaire de l'angle B du bras n NUMBRAS.

9 = >valeur courante de l'angle B du bras n NUMBRAS.

10 = >le type de liaison entre le bras n NUMBRAS et le bras n NUMBRAS - 1. Si la liaison est une rotule, XX_GET_INFO sera égale à 1; si la liaison est une charnière, XX_GET_INFO sera égale à 2.

11 = > les variations permises pour les angles A et B en fonction du type de liaison

Si A permis => XX_GET_INFO = 1
Si B permis => XX_GET_INFO = 2
Si A et B permis = >XX_GET_INFO = 3

12 = >une indication des axes le long desquels des translations sont permises. Il est à remarquer que pour ce code, il n'est pas nécessaire de spécifier le numéro du bras.

C'est l'ensemble du robot qui subit une translation.

Suivant X (non Y et Z) = >XX_GET_INFO = 1
Suivant Y (non X et Z) = >XX_GET_INFO = 2
Suivant Z (non X et Y) = >XX_GET_INFO = 3
Suivant X et Y (non Z) = >XX_GET_INFO = 4
Suivant X et Z (non Y) = >XX_GET_INFO = 5
Suivant Y et Z (non X) = >XX_GET_INFO = 6
Suivant X, Y et Z => XX_GET_INFO = 7

13 = dimension du bras NUMBRAS

Remarque: Pour faire appel à cette fonction, un robot doit évidemment exister, sans quoi, la fonction XX_GET_INFO retourne la valeur 0.

Restrictions :

0<CODE< 13.

NUMBRAS doit être au maximum égal au nombre de bras du robot, et au minimum à 0 (s'il existe un robot, il a toujours un organe actif).

Exemple :

1) data: =xx_get_info (1,1) L'exécution de cette fonction a pour effet d'assigner à la variable DATA la coordonnée x du bras n 1 dans le référentiel actif. A noter que la variable DATA est de type real.

2) data: =xx_get_info (14,2) L'exécution de cette fonction a pour effet d'assigner à la variable DATA une valeur indiquant les variations permises du bras n 2 en terme d'angle A et B. A noter que le bras n 2 existe.

5.2.3 Changement de point de vue.

Une propriété de l'interpréteur graphique est la possibilité donnée à l'utilisateur de changer le point de vue. Cette possibilité fut expliquée en long et large dans le chapitre consacré à cet effet. Dès lors, nous n'allons pas, dans ce chapitre, expliciter à nouveau cette caractérique. Mais voyons quels sont les moyens mis à la disposition de l'utilisateur désireux d'employer cette opportunité qui lui est offerte.

Deux moyens de changer le point vue sont présents. L'un requiert l'envoi d'une commande (garantissant la syntaxe vue ci avant et une certaine sémantique : cfr ci - après)et l'utilisation de touches du clavier, l'autre l'envoi d'une commande sans utiliser une touche quelconque. L'un deux par l'exécution de la procédure XX_SEND_COMMAND, l'autre par l'exécution de la procédure XX_OBSERVATION.

Lorsque la procédure XX_SEND_COMMAND est exécutée avec 'O' comme valeur du paramètre MOUV, l'utilisateur peut utiliser les différentes touches pour changer le point de vue : flèche-haut, flèche-bas, flèche-droite, flèche-gauche, pgup et pgdn. Il suffit d'appuyer sur ces différentes touches pour modifier l'angle de vision. Toute pression sur une touche autre que celles spécifiées ci-avant a pour effet de désactiver le clavier. Dans ce contexte, un autre paramètre est important, à savoir VALEUR qui indique le nombre de degrés dont l'angle de vision est modifié à chaque pression sur l'une des touches prévues à cet effet. VALEUR est un nombre entier compris entre 1 et 359. Par exemple, l'exécution de XX_SEND_COMMAND(O,?,?,?,5) met à la disposition de l'utilisateur les différentes touches utiles à la modification du point de vue et permet ainsi tout changement de l'angle de vision à chaque pression d'une touche adéquate. Il est à remarquer que les paramètres ANGLE, SIGNE et NUMBRAS n'interviennent dans l'éxécution de cette commande. Leurs valeurs peuvent dès lors être indéterminées.

Si l'utilisateur désire modifier l'angle de vue sans utiliser les touches du clavier, il fait appel à la procédure XX_OBSERVATION. Cette procédure s'accompagne de deux paramètres : AXE et VALEUR. Le paramètre AXE indique selon quel axe l'angle de vision est modifié et VALEUR indique l'ampleur de cette modification. Ainsi, si l'utilisateur désire changer le point de vue de 25 degrés selon l'axe X, il appellera XX_OBSERVATION (X,25).

Il faut insister sur le fait que l'utilisation de la procédure XX_OBSERVATION empêche totalement l'utilisateur de se servir du clavier pour modifier l'angle de vue. Les deux paramètres doivent répondre à certains critères :

- AXE doit avoir comme valeur X ou Y ou Z (le "ou" étant ici exclusif).
- VALEUR doit être = 1 et 359.

5.2.4 Que se passe-t-il en coulisse ?

Toute la communication se fait via une zone mémoire réservée à cet effet. Cette zone est commune aux deux programmes, l'interpréteur graphique et le programme créé par l'utilisateur.

Lorsque l'interpréteur graphique (le programme résident) s'exécute pour la première fois, il alloue de la mémoire en suffisance pour pouvoir mémoriser :

- les informations relatives aux bras du robot
- les informations relatives aux axes (utiles pour le changement de point de vue)
- les informations relatives aux origines des bras (utiles pour le calcul des coordonnées x,y,z, dans le référentiel écran, des différents bras du robot)
- une commande

De plus,il sauve, dans un fichier de type ASCII nommé "XX_COMM.DAT", le numéro d'interruption par lequel le programme de l'utilisateur pourra communiquer avec le simulateur ainsi que la valeur de segment de la zone mémoire réservée.

La mémoire de l'ordinateur, une fois la première exécution du programme résident terminé, présente la figure suivante :

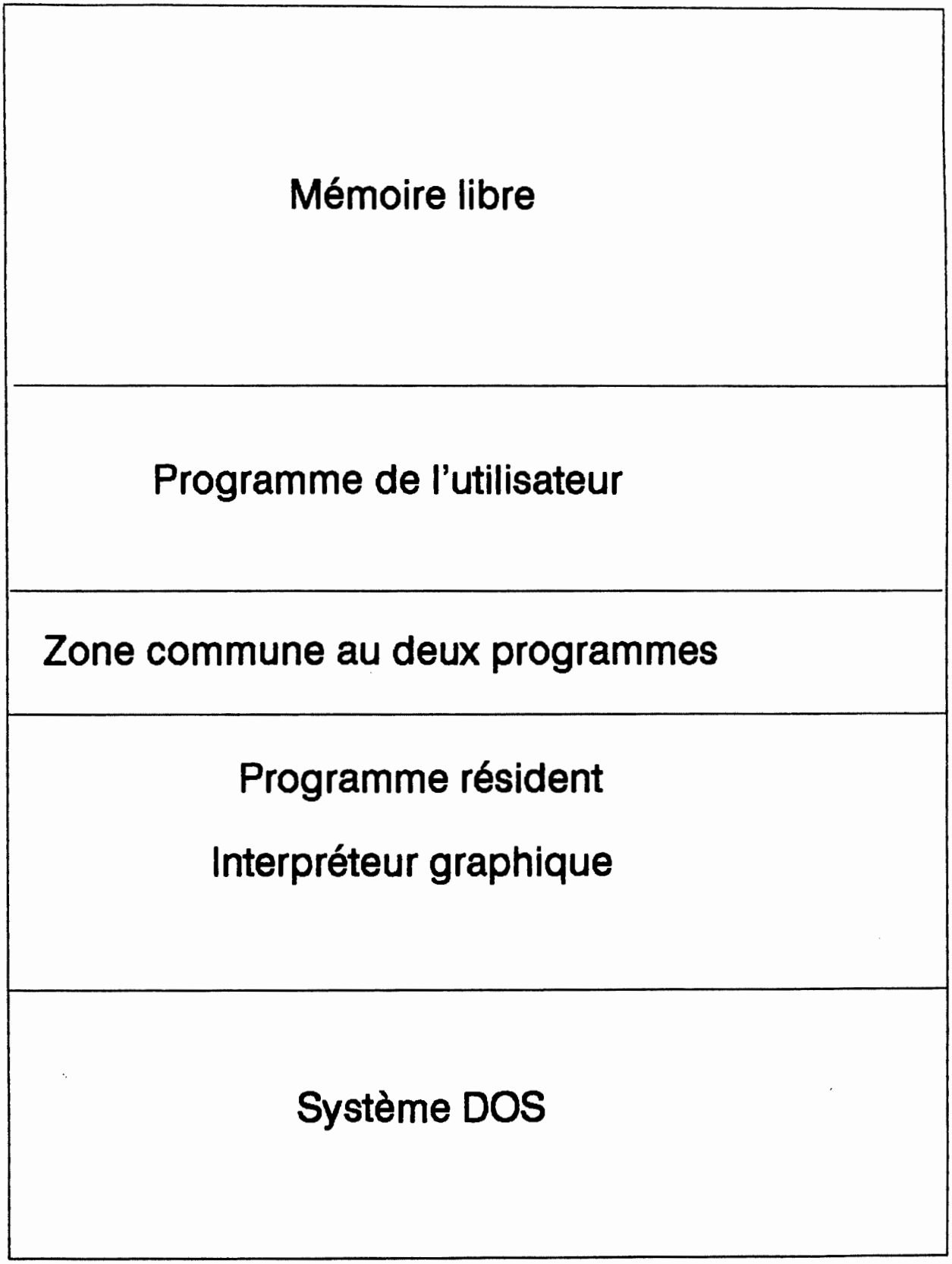
cfr dessin représentant le contenu de la mémoire page 5.19

Le programme de l'utilisateur s'exécutera au-delà de la zone mémoire allouée pour le stockage des informations communes au deux programmes. Tôt ou tard, il fera appel au différentes procédures servant d'interfaces de communication. Regardons de plus près le fonctionnement de toutes ces procédures et fonctions

5.2.4.1.XX_INITIALISATION

Lorsque XX_INITIALISATION est appelé par le programme de l'utilisateur, la première opération est d'aller lire dans le fichier le numéro d'interruption et la valeur du segment où la zone de mémoire réservé au stockage d'information débute. Connaissant le numéro d'interruption, il peut communiquer avec l'interpréteur graphique résident en mémoire afin d' initialiser le mode graphique. Il suffit pour cela d'exécuter l'interruption en ayant au préalable assigné une valeur appropriée dans le registre AX. En effet, la procédure d'interruption au sein de l'inter-

Contenu de la mémoire



préteur graphique présente comme structure une opération "CASE "(sens Pascal) sur base du registre AX. De cette façon, il est aisé de savoir si le programme appelant est interrompu pour l'initialisation (initialisation du mode graphique, chargement des fichiers textes, ...), ou pour le chargement ou la construction d'un robot, ou pour l'interprétation d'une commande, ...

L'initialisation du mode graphique est effectuée si l'interruption réservée par le programme résident est exécutée avec la valeur 1 dans le registre AX.

En plus de l'initialisation, l'interpréteur répartit l'espace écran en déterminant les bornes respectives à chacun des programmes et les sauve dans le fichier XX_COMM.DAT. Au retour, la deuxième opération de XX_INITIALISATION consiste à activer le fenêtrage en fonction des bornes lues dans le fichier XX_COMM.DAT.

5.2.4.2. XX_MADE_ROBOT

Cette procédure est chargée d'appeler la partie de l'interpréteur graphique résident en mémoire responsable de la construction ou du chargement d'un robot. L'interruption adéquate est exécutée en ayant comme valeur dans le registre AX la valeur 2.

5.2.4.3 XX_SEND_COMMAND

Cette procédure est appelée par le programme de l'utilisateur désireux de transmettre une commande à l'interpréteur graphique. Avant d'exécuter l'interruption réservée par le programme résident avec le registre AX égal à 3, il faut stocker la commande dans une partie de la zone mémoire réservée par l'interpréteur graphique. Mais quelle la partie qui est réservée à cette fonction ?

Faisons d'abord un petit rappel du mécanisme d'accès à un emplacement de la mémoire. Nous verrons ensuite comment se structure la zone mémoire réservée par l'interpréteur graphique.

5.2.4.3.1. Emplacement mémoire et mécanisme d'adressage.

L'unité minimale de mémoire, encore appelé emplacement mémoire, est le byte (8 bits). Un byte d'information est suffisant pour mémoriser n'importe quel caractère alphabétique, numérique et de ponctuation plus un ensemble de caractères plus ou moins graphique. La lettre "A" requiert, par exemple, 1 seul byte de mémoire. Une valeur entière peut varier de -32768 à 32767, ce qui requiert 2 bytes.

Valeurs entières.

Binaire	Décimal
1000 0000 0000 0000	-32768
1000 0000 0000 0001	-32767
...	...
0111 1111 1111 1110	32766
0111 1111 1111 1111	32767

Il se peut que des objets requièrent non pas un byte mais plusieurs bytes comme c'est déjà le cas pour une valeur entière. Examinons quel est le nombre de byte(s) requis en fonction du type de donnée. Etant donné que le langage de haut niveau utilisé pour la programmation de l'interpréteur graphique est le Pascal, plus particulièrement Turbo Pascal version 5, nous examinerons les différents types de données (ceux que nous avons utilisés) qu'offre un tel produit.

Type de donnée	Valeur	Taille en bytes
integer	- 32768 -> 32767	2
longint	-2147483648 -> 2147483647	4
real	10 E -38 -> 10 E 38	6

Chaque emplacement mémoire (byte) doit posséder une et une seule adresse. Les IBM PC et compatibles assurent cette pseudo - propriété en utilisant un mécanisme d'adressage basé sur les notions de segment, d'offset et de paragraphe. La mémoire est découpée en paragraphes de 16 bytes. Chaque paragraphe possède une adresse : le segment. Il est possible d'accéder à un byte à l'intérieur d'un paragraphe par l'offset. Le format d'adressage à un byte est le suivant: mem(segment:offset)

5.2.4.3.2. Zone de mémoire réservée par l'interpréteur graphique : structure.

Cette zone mémoire est utile pour stocker les informations sur le robot, sur les axes du référentiel, ... (cfr infra). Cette zone mémoire fut réservée en exécutant le service DOS n 48H (hexadécimal).

Service Dos 48H
Fonction: Allocation de paragraphe(s) mémoire(s).

Entrée

Registre

AH <- 48h : service d'allocation de paragraphe(s) mémoire(s).

BX <-Nombre de paragraphe(s) mémoire(s) désiré(s)

Sortie

Registre

AX <- Code d'erreur si le carry flag est mis à 1. Si pas d'erreur dans la requête de paragraphes, l'adresse du segment du bloc mémoire alloué est mémorisé dans AX.

BX<- Taille du plus gros bloc mémoire disponible si l'espace mémoire demandé ne peut être satisfait étant donné l'inexistence d'un tel espace mémoire contigu.

La routine utilisée se présente comme suit:

```
procédure allocation_memoire (paragraphe : integer;        var segment,size : integer);
type regs : record
              ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
       end;
var r : regs;
begin
with r do
       begin
              ax:=$4800;
              bx:=paragraphe;
              msdos (r); (* appel au service Dos *)
              segment:=ax;
              size:=bx;
       end;
end;
```

Un robot est composé d'1 maximum de 5 bras plus un organe actif. Chaque bras est composé d'un axe directeur et de deux axes secondaires. Chaque axe stocke 17 data différentes, toutes étant de type real. Cela signifie que chaque axe requiert : 17 (data) * 4 (type d'une data : real = 4 bytes) = 68 bytes, ce qui donne 5 paragraphes utiles (5*16=80).

Or, il y a 5 bras et un organe actif,ce qui fait 18 axes. Il faut donc 18 * 5 = 90 paragraphes pour stocker les informations sur les différents bras du robot.

Pour les données constituantles origines de chaque bras et utiles lorsqu'il s'agit de calculer les coordonnées x,y et z des bras du robot dans le référentiel écran, examinons la place mémoire requise. Pour chaque bras, il existe un groupe de 6 données, ce qui demande 6 * 4 (real) = 24 bytes, autrement dit 2 paragraphes (2 * 16 = 32). Or, il existe 6 bras (5 + 1 organe actif); il faut donc 12 paragraphes.

En ce qui concerne les informations utiles au changement de point de vue, il faut compter que chaque axe nécessite 8 data de type real, ce qui donne 8 * 4 = 32 bytes, autrement dit 2 paragraphes. Le référentiel est composé de 3 axes. Il faut 3 * 2 = 6 paragraphes pour mémoriser les informations relatives au changement de l'angle de vision.

Il est nécessaire de réserver à nouveau un paragraphe qui servira de zone de communication de commandes entre les deux programmes. Il faut se rappeler qu'une commande est composée de 5 paramètres :

- nature de la commande (MOUV) : 1 data de type caractère.
- angle ou axe concerné par le mouvement (ANGLE) : 1 data de type caractère.
- signe du mouvement (SIGNE) : 1 data de type caractère.
- le numéro du bras (NUMBRAS) : 1 data de type integer.
- l'ampleur du mouvement (VALEUR) : 1 data de type integer.

En résumé, il faut compter que la zone mémoire qui sera réservée aura la taille suivante :

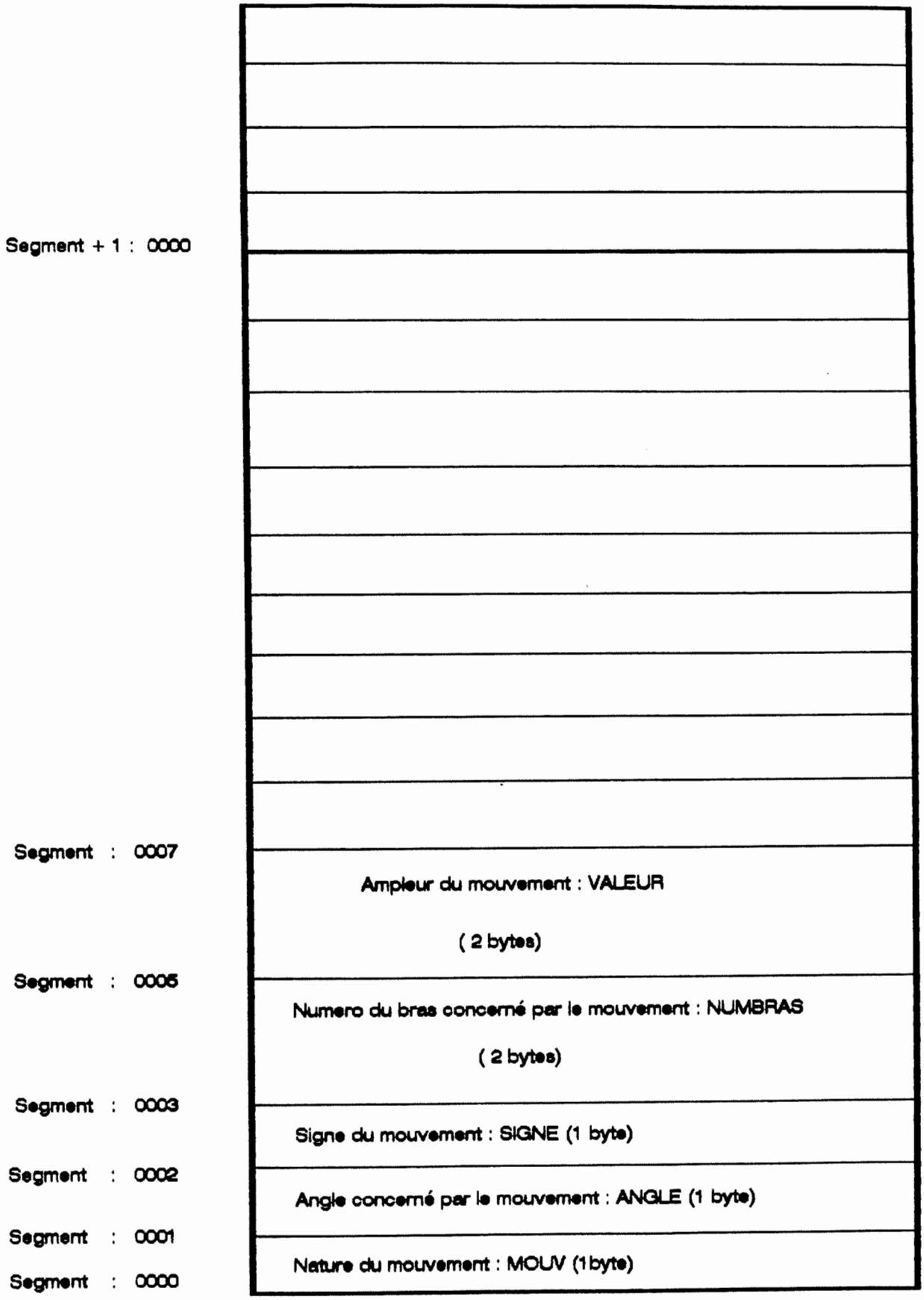
data relative au robot	90 paragraphes
" " aux origines	12 paragraphes
" " à la vision	6 paragraphes
" " à 1 commande	1 paragraphe
TOTAL	109 paragraphes.

Structure réservée pour le stockage d'un axe de données relatives au robot. : cfr page 5.24

Structure réservée pour le stockage de données relatives au passage d'une commande. : cfr page 5.25

SEGMENT OFFSET	
segment + 4 : 000C	
segment + 4 : 0008	
segment + 4 : 0004	
segment + 4 : 0000	Axe(s) de translation permis (4 bytes)
segment + 3 : 000C	Variation permises en terme d'angle A ou /et B (4 bytes)
segment + 3 : 0008	Type de liaison (4 bytes)
segment + 3 : 0004	Information relative au point d'attache (4 bytes)
segment + 3 : 0000	Dimension du bras (4bytes)
segment + 2 : 000C	Valeur courante de l'angle B (4 bytes)
segment + 2 : 0008	Valeur intermédiaire de l'angle B (4 bytes)
segment + 2 : 0004	Valeur initiale de l'angle B (4 bytes)
segment + 2 : 0000	Valeur courante de l'angle A (4 bytes)
segment + 1 : 000C	Valeur intermédiaire de l'angle A (4 bytes)
segment + 1 : 0008	Valeur initiale de l'angle A (4 bytes)
segment + 1 : 0004	Coordonnée Z dans le référentiel écran (4 bytes)
segment + 1 : 0000	Coordonnée Y dans le référentiel écran (4 bytes)
segment : 000C	Coordonnée X dans le référentiel écran (4 bytes)
segment : 0008	Coordonnée Z dans le référentiel actif (4 bytes)
segment : 0004	Coordonnée Y dans le référentiel actif (4 bytes)
segment : 0000	Coordonnée X dans le référentiel actif (4 bytes)

SEGMENT OFFSET



5.2.4.4 XX_GET_INFO

Cette fonction est appelée par l'utilisateur désireux d'avoir des informations sur le robot, que ce soit les coordonnées x,y ou z dans le référentiel actif, la valeur courante de l'angle A .... Connaissant le segment du début de la zone mémoire commune aux deux programmes et selon un algorithme qui permet de trouver l'adresse-mémoire (segment + offset) de l'information demandée, l'information est trouvée et fournie à l'utilisateur.

Algorithme de calcul d'adresse.

1) Recherche du segment du paragraphe.

La formule utilisée pour déterminer avec précision le segment du paragraphe où débute l'information relative au bras n Numbras est la suivante :

segment=(segment_debut_de_zone+((numbras-1)*15)) (1)
+(num_colonne*4)div 16 (2)

Quelques explications :

(1) Un bras est formé de 3 axes (1 directeur et 2 secondaires). Parmi ces 3 axes, seul l'axe directeur importe pour fournir l'information demandée; les axes secondaires étant utilisés pour dessiner le bras à l'écran. En sachant qu'un axe occupe 5 paragraphes de la mémoire, 3 en occupent 15. Dès lors, il faut aller de 15 en 15 paragraphes. Num_colonne indique l'élément demandé(cfr chapitre sur les concepts; definition du robot). Sa valeur est fonction du code donné.

Si 0 < code < 4 alors num_colonne = code
Si 3 < code < 10 alors num_colonne = code + 3
Si 9 < code < 12 alors num_colonne = code + 5;

Numbras est le numéro de bras choisi pour lequel de l'information est demandée. Pourquoi "Numbras - 1" ? L'information relative au premier bras est mémorisée au paragraphe dont l'adresse est le segment "segment_debut_de_zone" et non 15 paragraphes plus loin, ce qui serait le cas si la valeur de Numbras était seulement prise en considération. Par exemple, Numbras valant 1, l'information relative à ce bras commence bien,si la formule est appliquée, au paragraphe dont l'adresse est"segment_debut_de_zone".

(2) Num_colonne intervient dans le calcul du segment du paragraphe étant donné que l'information d'un axe s'étend sur 5 paragraphes. Il se peut que l'information demandée se situe dans le 3ème paragraphe.

Num_colonne est multiplié par 4 étant donné que l'information mémorisée est de type real (4 bytes).

2)Recherche de l'offset à l'intérieur du paragraphe.

La formule est toute simple :

offset = (num_colonne*4) modulo 16;

5.2.5 Avantage d'utiliser l'interface de communication

L'utilisation de ces différentes procédures et fonctions permet à l'utilisateur d'ignorer la présence de cette zone mémoire ainsi que sa structure. Il est également possible de modifier la communication entre les deux programmes sans pour autant changer le libellé des procédures et fonctions. La maintenance est d'autant plus facilitée. Ceci ne représente pas l'utilisation proprement dite de la notion de type abstrait mais s'y rapproche fortement.

Dans le cas où un concepteur désirerait faire évoluer la communication entre les deux programmes, il doit modifier deux unités de l'ensemble. L'unite XXOUTIL et TABLEAUX. XXOUTIL offre les outils utiles à la communication alors que TABLEAUX représente une librairie de procédures et fonctions qui permettent d'accéder aux différentes structures de données. Le concepteur sera amené à modifier uniquement le corps de ces procédures et fonctions sans pour autant apporter des changements dans d'autres unités.

Chapitre 6

Conclusion

CONCLUSION

Le logiciel développé implémente tous les éléments nécessaires afin d'assurer une simulation correspondant relativement bien au comportement d'un robot observé dans la réalité. Ainsi, l'implémentation d'une majorité des commandes (seule la saisie d'un objet par l'organe actif n'est pas traitée dans l'état actuel du système) rend le système suffisamment puissant pour traiter tous les robots répondant aux caractéristiques spécifiées.

De même, en ce qui concerne l'évaluation de la simulation, elle est rendue possible par la représentation de la trajectoire sous différentes manières : visualisation par liaison par des segments de droite des points de passage, affichage à l'écran et sur papier des coordonnées de ces points, ... Cela permet de faire facilement une analyse des résultats obtenus.

La représentation de la notion de perspective à partir de celle de grosseur assure une présentation de la simulation de manière plus conforme à la réalité dans la mesure où cette particularité est associée à des caractéristiques des robots présents dans la réalité. En effet, un bras de robot proche de l'observateur semblera toujours plus gros qu'un bras éloigné. De même, pouvoir observer un robot à partir de différents points de vue doit permettre une meilleure évaluation de la position du robot tant dans la réalité que lors de la simulation.

Le problème de la transmission des commandes à partir d'un programme auxiliaire est également un élément important du logiciel. En effet, ce point permet l'exécution tant de façon "ON LINE" que "OFF LINE" d'un ensemble de commandes. Cela correspond aux différentes programmations d'un robot. La transmission des commandes de façon interactive à partir du programme de commandes correspond assez bien à la programmation "ON LINE" tandis la création d'un fichier batch de commandes peut être assimilée à une programmation "OFF LINE" du robot.

Nous pouvons de ce fait dire que le logiciel répond aux besoins exprimés en vue de définir un outil informatique afin d'aider les ingénieurs en robotique dans leur tâche de résolution de programme pour des robots.

Chapitre 7

Propositions

7. PROPOSITIONS D'ADAPTATIONS

Le logiciel, dans l'état actuel de développement, présente un certain nombre de limites. La priorité a été donnée à l'implémentation d'un maximum de concepts relatifs à un robot simple au détriment de l'implémentation pour un robot plus complexe. Les limites du logiciel ont principalement trait à la simplicité du robot modélisé.

Le logiciel se limite également à la visualisation d'un certain nombre de commandes fournies à un robot sans faire d'interprétation sémantique de ces commandes ni des résultats fournis. Des propositions sont faites afin de pouvoir corriger facilement les limites et lacunes du logiciel actuel.

Ce chapitre relève un certain nombre d'aménagements susceptibles de l'adapter à un grand nombre de robots aussi complexes que possible mais également d'apporter un certain nombre d'améliorations et d'options supplémentaires. Ainsi des propositions en vue de compléter l'analyse du comportement du robot (interprétation sémantique des commandes et des résultats,...) sont présentées dans la suite de ce chapitre.

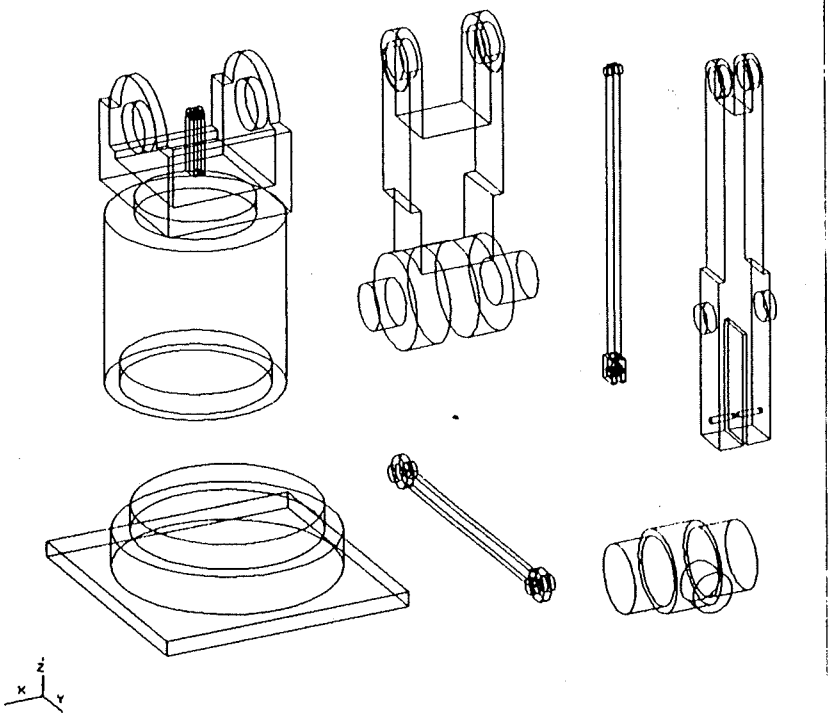
7.1. CONSTRUCTION DU ROBOT

Une alternative à notre méthode de construction du robot qui subit la simulation consiste en l'affichage de tous les objets susceptibles de constituer le robot.

L'utilisateur a alors la possibilité de choisir l'élément qu'il désire ajouter à la structure actuel du robot.

Pour faciliter cette manière de faire, l'usage de la souris serait particulièrement adapté.

L'implémentation logique de cette méthode de construction du robot se rapproche assez bien de la méthode utilisée actuellement dans le logiciel. En effet seul l'interface avec l'utilisateur serait modifiée, le principe restant toujours l'affichage d'écran avec sélection des choix souhaités.

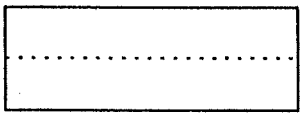


2. VISUALISATION DU ROBOT

Chaque bras constitutif de la structure du robot est représenté par des segments de droite.

Dans le but d'améliorer la visualisation du robot, cette représentation peut être réalisée à partir de faces agencées de telle manière que le bras apparaisse en 3 dimensions à l'écran.

Pour implémenter cette solution, seule la partie DESSIN doit être modifiée. Il est ainsi nécessaire d'adapter ce module afin qu'il puisse dessiner un bras en 3 dimensions, toujours à partir des éléments définissant chaque bras du robot tout en tenant toujours compte de la notion de grosseur du bras. Ce module, au lieu de manipuler des segments de droite, manipulerait des quadrilatères construits en fonction du contenu du tableau du robot et les agencerait de manière à représenter un bras en 3 dimensions.



Représentation actuelle

Représentation en 3D

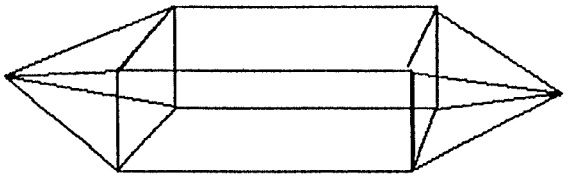


Fig 6.1 Visualisation du robot : proposition

3. INTREPRETATION SEMANTIQUE DES COMMANDES

Dans l'état actuel du système, aucune interprétation sémantique des commandes n'est faite. Les commandes sont fournies selon un format et sont supposées syntaxiquement correctes.

Cependant, chaque commande peut être visualisée à l'écran sans savoir si elle est acceptable pour un robot réel. Ainsi, deux bras pourraient être placés de telle manière qu'ils soient superposés alors que, dans la réalité, cette situation est impossible à cause des contraintes matérielles.

Pour un robot réel, des contraintes physiques existent et limitent ainsi certaines commandes. De ce fait, des bornes peuvent être, dans la réalité, imposées pour certains mouvements et certaines positions peuvent être impossibles.

Pour traiter l'analyse du caractère acceptable de la position résultant d'un mouvement, un module pourrait s'en charger. Il aurait pour effet de calculer au préalable cette position et ses implications et de tester si elle est acceptable. Si c'est le cas, la commande est visualisée tandis que dans le cas contraire, elle est rejetée.

Un point particulier de cette interprétation pourrait permettre le développement d'une commande qui aurait pour but de vérifier si la saisie d'un objet est réalisable ou non en fonction des contraintes de forme et de poids. Le module d'interprétation serait alors également chargé d'obtenir tous les renseignements concernant l'objet à saisir.

7.4. INTERPRETATION DES RESULTATS

Une suite intéressante qui pourrait être donnée à ce logiciel est l'ajout d'un module qui, en fonction de la position résultante du dernier mouvement, proposerait un ensemble de commandes possibles qui auraient pour effet de faire évoluer le robot vers une position finale, préalablement définie.

Ce point ferait assez bien le lien entre le programme de simulation et un système expert adjoint à ce programme.

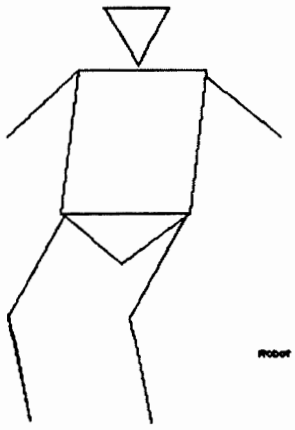
Pour implémenter cette caractéristique, on peut toujours utiliser les éléments actuels puisu'ils permettent d'obtenir la position courante du robot et assurent sa visualisation

7.5. ROBOT AYANT PLUSIEURS POINTS FIXES

Le logiciel se limite à la simulation du comportement de robots ayant un seul point fixe. Cette limite a été imposée afin de diminuer la complexité de gestion déjà importante au départ. En effet, la priorité a été donnée à l'analyse et la gestion complète de toutes les caractéristiques nécessaires pour avoir la simulation d'un robot relativement simple (affichage de la trajectoire, notion de perspective, mémorisation du robot,...).

Le logiciel dans l'état actuel de développement ne permet pas le traitement de robot aussi complexe que l'on veut. Quelques propositions sont faites maintenant pour étendre l'utilisation de ce logiciel à d'autres robots.

Pour résoudre ce problème, il est conseillé de se référer à l'article de BIRTWISTLE [G.BIRTWISTLE] où une démarche de résolution y est présentée. Cette démarche est particulièrement bien adaptée au traitement de robot sensé représenter un être humain.



Robot ayant plusieurs points fixes

Fig 4.2

7.6.ROBOT POUVANT AVOIR PLUSIEURS ORGANES ACTIFS

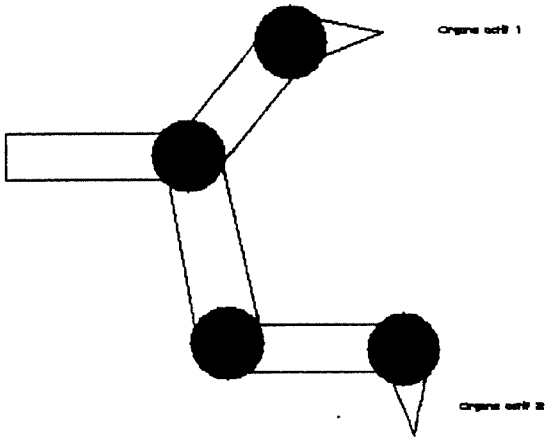
Le logiciel se limite également à des robots ayant un seul bras jouant le rôle d'organe actif. Cette limite se justifie également de la même manière que celle résultant d'un seul point fixe.

Pour qu'un robot puisse avoir plusieurs organes actifs, il est nécessaire qu'à au moins un endroit, il y ait deux ou plusieurs bras qui soient attachés à un autre bras.

Ce type de robot demande la création d'un module particulier chargé d'identifier sans équivoque chaque bras du robot.

Les mouvements restent les mêmes lorsque l'identification des bras a été réalisée.

La saisie d'objet est également possible à partir du moment où le module chargé de tester si la saisie est faisable a été implémenté.



Simulation quantitative en robotique

Partie 2 et 3

E. CONRARD D. DARTE

Mémoire présenté

en vue de l'obtention du titre

Licencié et Maître en sciences

option Informatique

Septembre 1989

Institut d'informatique des Facultés Notre-Dame de la Paix de Namur

Chapitre 8

Lancement

8. LANCEMENT DU PROGRAMME

Dans ce chapitre, nous vous présentons les instructions pour installer le programme et les outils auxiliaires sur un disque dur ou sur une autre disquette. Nous vous informons également du contenu de la disquette ainsi que sa structure. Enfin, nous vous décrivons les étapes à suivre pour employer le simulateur de robot

8.1 Installation sur disquette.

La disquette de distribution qui accompagne ce manuel est formatée en 360 k , de 5.1/4 (inch), et peut être lue par des IBMPC ou autres compatibles. Mais avant de faire quoique ce soit, il est utile que vous fassiez un back-up de la présente disquette, et une fois cette opération réalisée, garder l'original intact. En effet, au moindre dommage occasionné par votre copie, vous pouvez utiliser l'original pour faire à nouveau un back-up et uniquement un back-up.

Voici le processus à suivre :

·Disposer d'une nouvelle disquette

· Rebooter votre ordinateur

·Au prompt, tapez "diskcopy A : B :" et confirmer en appuyant sur la touche ENTER ou RETURN. Le message suivant : "Insert source diskette in drive A :" sera affiché à l'écran. Enlevez votre disquette système du drive A et introduisez le disque de distribution dans le drive A.

· Si votre système dispose de 2 drives, votre écran affichera également : "Insert destination diskette in drive B".Dans ce cas, vous devez retirer la disquette présente dans le drive B et y placer votre disquette vierge. Si votre système dispose seulement d'un seul drive, vous êtes dans l'obligation de permuter un certain nombre de fois la disquette source avec la disquette destination dans le drive A. Rappelez-vous simplement que la disquette de distribution est la disquette source, la disquette vierge est la disquette destination.

· Si vous ne l'avez déjà fait, appuyer sur "Enter". L'ordinateur commencera la lecture de la disquette source introduite dans le drive A.

· Si vous avez un ordinateur avec 2 drives, il écrit sur la disquette destination se trouvant dans le drive B et continue à lire sur la disquette du drive A et écrire sur la disquette du drive B jusqu'au moment où la copie sera réalisée. Si vous avez un seul drive, il vous est demandé de placer la disquette destination dans le drive A, puis la disquette source, puis la disquette destination, et ainsi de suite jusqu'à la fin du back-up.

· Quand la copie est terminée, enlevez la disquette de distribution du drive A et mettez de côté.

8.2. Installation sur disque dur

L'installation de l'ensemble des programmes présents sur la disquette distribution passe par l'utilisation d'un programme d'installation spécialement conçu pour cette tâche.

Pour lancer l'exécution de ce programme, suivez les étapes suivantes :

· Robooter votre ordinateur

· Introduisez la disquette de distribution dans le drive A

· Au prompt, tapez : "cd A:simrob" et confirmez en appuyant sur la touche "Enter" ou "Return"

· Frappez :"Instalhd', et confirmez par un appui sur la touche "Enter" ou "Return". Vous êtes dans le programme d'installation. Il vous est demandé si vous voulez installer le package de programmes sur disque dur.

· Si vous désirez répondre par la négative, frappez la touche "n" ou "N". Dans ce cas, vous sortez du programme d'installation. Dans le cas contraire, vous êtes soumis à une autre question

· Quel est le directory dans lequel vous voulez installer le package des programmes. Il existe un directory, par défaut, à saisir : "c:/simrob". Dans le cas où vous acceptez ce ss-directory, il suffit de taper "O" suivi d'une confirmation par la frappe de la touche "Enter"ou "Return".

· Si vous ne désirez pas installer l'ensemble des programmes dans ce ss-directory, tapez "N" ou "n" de sorte que vous vous voyez invité à introduire le ss-directory dans lequel le programme d'installation copie tous les programmes de la disquette distribution.

· Attention, lors de l'introduction du nouveau ss-directory d'installation, les mêmes règles qu'au niveau du Dos sont à respecter. Par exemple, le nombre de caractères composant le nom d'un ss-directory ne peut pas être supérieur à 8. Par contre, le nombre maximum de caractères du chemin pour arrivé à ce sous-directory est limité à 74. C'est la seule restriction par rapport au DOS.

8.3. Contenu du disque de distribution

Le disque de distribution contient un package fichiers de programmes. L'ensemble de ces fichiers de programmes sont nécessaires à l'exécution du programme principal. En plus de ces fichiers, une collection de fichiers textes est présente. Tous ces fichiers, textes et programmes, sont répartis selon une structure de directory.

Nous pouvons relever la structure suivante. Décrivons le contenu de chaque directory.

SIMROB

Ce ss-directory contient l'ensemble des fichiers exécutables :
· resident exe : Ce programme est le simulateur proprement dit
· config exe : ce programme permet de fixer un environnement hardware dans lequel simrob.exe s'exécute ainsi que de déterminer une interface conviviale pour l'utilisateur

· *. bgi : l'ensemble de ces fichiers sont utiles à l'installation du mode graphique

· XX-outil.* : l'ensemble des fichiers utiles lors de la création d'un programme par utilisateur.

Français

Ce ss-directory de Simrob contient les fichiers textes français de tous les écrans qui seront utilisés par les différents programmes (simrob, config)

Anglais

Ce ss-directory de Simrob contient les fichiers textes anglais de tous les écrans qui seront utilisés par les différents programmes (simrob exe, config exe)

@RET 0 CM 80 C = Portugais

Ce ss-directory de Simrob contient les fichiers textes portugais de tous les écrans qui seront utilisés par les différents programmes (simrob exe, config exe)

Allemand

Ce ss-directory de Simrob contient les fichiers textes allemands de tous les écrans qui seront utilisés par les différents programmes (simrob exe, config exe)

Utilis

Ce ss-directory ne contient rien initialement. Ce n'est qu'après avoir exécuter le programme simrob exe que vous pourrez remarquer un certain contenu. En aucun cas, ne supprimez ce ss-directory. Sa suppression pourrait être fatale à l'exécution de simrob exe.

Data

Ce ss-directory présente les mêmes caractéristiques que le ss-directory UTILIS. Dès lors, il faut en aucun cas le supprimer.

Instalhd.

Ce ss-directory contient l'ensemble des fichiers exécutables et textes.

- Instalhd-exe : Programme d'installation surdisque dur
- *. txt : Ces fichiers textes contiennent tous les écrans qui seront affichés lors de l'exécution de Instalhd.

Remarque : Ces textes sont disponibles uniquement en français.

8.4 Lancement du programme.

Le simulateur ne sait pas s'exécuter seul. Il est utilisé par un programme auxiliaire créé par un utilisateur. Ce programme est responsable de la saisie et de la communication des commandes au simulateur.

Pour lancer l'exécution de ces 2 programmes, il suffit de taper, si les deux programmes se trouvent dans le même sous-directory :

SIMULROB UTILISAT

où utilisat est le programme auxiliaire.

Il est à remarquer que les deux programmes doivent se trouver dans le même sous-directory,sans quoi, il n'est pas possible de lancer l'exécution.

8.5 Restriction.

La présence d'un disque virtuel est une obligation. Il suffit d'ajouter dans le fichier CONFIG.SYS de votre système une commande du genre : DEVICE = RAMDRIVE.SYS. Pour plus de renseignements, consulter votre manuel de DOS.

Chapitre 9

Manuel

9.MANUEL D'UTILISATION DETAILLE

Ce manuel d'utilisation est un outil qui doit permettre à tout utilisateur de pouvoir faire exécuter le programme mis à sa disposition de manière correcte et optimale. Il s'adresse à tout utilisateur, qu'il soit novice ou qu'il soit expert. Cette partie se limite cependant au fonctionnement du programme RESIDENT et du programme configuration, le lancement du programme ayant été présenté précédemment.

Il a pour but d'expliquer la marche à suivre en vue d'une exécution correcte du programme SIMROB ainsi que la justification de toutes les erreurs commises et la correction nécessaire avant de poursuivre sa session de travail.

La première partie a pour objet la description de la saisie des données à partir du clavier.

Dans la deuxième partie, il y a une présentation des moyens mis à la disposition de l'utilisateur pour qu'il puisse construire le robot dont il désire simuler le comportement.

Dans la troisième partie, on spécifie la procédure à suivre pour la mémorisation d'un robot dans le but de pouvoir le retrouver lors d'une simulation ultérieure.

La quatrième partie explique le contenu du fichier de configuration adjoint au programme RESIDENT et les possibilités de le modifier.

La description des commandes et la marche à suivre pour les faire exécuter à partir du programme auxiliaire sont présentées dans le chapitre 4.

Les écrans auxquels on se réfère lors de la description des différentes étapes de l'exécution du programme sont présentés dans les annexes.

9.1. SAISIE ET CORRECTION DES DONNEES

Au cours de certaines étapes de sa session de travail, l'utilisateur est amené à saisir un caractère ou une suite de caractères (Les données numériques sont traitées comme une suite de caractères). Toutes les données saisies sont constituées à partir des caractères alphanumériques qui sont introduits à partir du clavier.

9.1.1 SAISIE ET CORRECTION D'UN CARACTERE

Lors de certaines étapes, l'utilisateur doit introduire une réponse sous la forme d'un seul caractère à lire à partir du terminal avant de pouvoir passer à l'étape suivante.

Lorsque l'utilisateur doit introduire une réponse à une telle question (réponse par OUI ou par NON le plus souvent), il lui faut frapper le caractère correspondant. Les réponses possibles lui sont indiquées de manière bien visible et toutes réponses autre que celles proposées lui sont interdites. Ces réponses incorrectes sont d'ailleurs rejetées comme étant incorrectes et ne sont pas affichées à l'écran. Cela évite de devoir les effacer avant de saisir une des réponses correctes.

Tant qu'il n'y a pas eu confirmation de la réponse saisie, l'utilisateur garde la possibilité de modifier son choix en effaçant le choix précédent et en recommençant la saisie du caractère correspondant à la réponse souhaitée. Pour effacer son choix précédent, il dispose de la touche DEL prévue à cet effet.

Pour continuer sa session, l'utilisateur doit confirmer son choix. Pour ce faire, il doit utiliser la touche ENTER prévue à cet effet.

Notons que même après avoir confirmé son choix, l'utilisateur garde toujours la possibilité de remonter à l'étape précédente ou à l'étape initiale. Cependant en revenant en arrière lors d'une étape, tout ce qui avait été saisi lors de l'étape courante est perdu. Cet inconvénient pourrait être éliminé en passant à chaque fois par un écran qui aurait pour but d'avertir des conséquences de la réponse et de demander de confirmer une deuxième fois le choix réalisé. Cette procédure a l'inconvénient d'être relativement désagréable pour un utilisateur suffisament averti et allonge sensiblement le déroulement des étapes. Cette possibilité a cependant été gardée pour une étape particulière. En effet, à partir du moment où la lecture du robot que l'on souhaite utiliser a débutée, une mauvaise manipulation pourrait faire perdre tout ce qui a déjà été saisi et il est évidemment assez fastidieux pour l'utilisateur de devoir recommencer toute la lecture en entier. Pour cette seule étape, il a ainsi été prévu une deuxième confirmation avec prévention des conséquences du choix.

9.1.2 SAISIE ET CORRECTION D'UNE SUITE DE CARACTERES

Lors de certaines étapes, l'utilisateur peut être amené à saisir une ou plusieurs suites de caractères, chaque suite de caractères correspondant à un champs à saisir.

Le champs courant est indiqué par le caractère '*' placé avant le libellé du champs qui est en cours de saisie.

La saisie de la suite de caractères se fait à partir du clavier par la frappe successive de tous les caractères qui doivent former le champs saisi.

Lors de la saisie du champs courant, dès que le premier caractère a été introduit, le caractère '*' est également placé avant la suite saisie. Ce caractère reste placé tant que la saisie n'a pas été confirmée. Il a pour but d'indiquer que la saisie de ce champs est en cours et qu'elle n'est pas terminée puisque la confirmation n'a pas encore été réalisée. Celle-ci se fait en utilisant la touche ENTER. La confirmation a pour effet d'effacer les caractères '*' placés aux deux endroits et de désigner le champs suivant comme étant le champs courant. Lorsque le dernier champs est atteint, la confirmation laisse ce champs comme étant le champs courant tout en effaçant le caractère '*' placé avant la suite de caractères saisis.

L'utilisateur n'est pas forcé de saisir les champs dans l'ordre dans lesquel ils se présentent. Il peut en effet se déplacer d'un champs à l'autre en utilisant les touches 'flèche haut' et 'flèche bas'. Cependant, dès que la saisie d'un champs est commencée (un ou plusieurs caractères ont été saisis), il n'est pas possible de passer à un des champs suivants ou précédents sans avoir confirmer la saisie du champs commencée.

Pour corriger un champs dont la saisie n'a pas été confirmée, on dispose de la touche DEL prévue à cet effet et qui provoque l'effacement de tous les caractères saisis jusqu'à ce moment. La saisie de nouveaux caractères peut alors commencée. On considère que, pour ce champs, la saisie est commencée même si aucun caractère n'a pas encore été saisi depuis l'effacement. Pour passer à un autre champs, il faut donc nécessairement confirmer le champs en cours de saisie.

Pour corriger un champs dont la saisie a été confirmée, il suffit de se déplacer jusqu'à ce champs et de recommencer la saisie sans tenir compte de la saisie précédente de ce champs. La frappe du premier caractère a pour effet, non seulement, de placer le caractère '*' avant le champs saisi mais également d'effacer tout ce qui avait été introduit précédemment pour ce champs.

Pour passer à l'étape suivante, il suffit d'utiliser la touche adéquate indiquée au bas de l'écran. Le passage à cette étape suivante n'est possible que si le dernier champs saisi a été confirmé. Avant de passer à l'étape suivante, une validation des données est réalisée. Cette validation s'arrête dès qu'elle a détecté une donnée invalide.

Si toutes les données sont valides, on passe à l'étape suivante. Dans le cas contraire, un message qui explique l'erreur commise est affiché au bas de l'écran . Pour continuer, il faut alors frapper la touche ENTER, qui a, dans cette situation, pour effet d'effacer le message et de se placer sur le premier champs non-valide; une nouvelle saisie de cette donnée est alors permise et le processus se répète tant que toutes les données ne sont pas valides.

9.1.3 SELECTION D'UN CHAMPS SANS SAISIE.

Lors de certaines étapes, l'utilisateur doit choisir une réponse parmi celles qui lui sont présenté à l'écran.

Pour ces étapes, il serait trop contraignant et trop long de demander à l'utilisateur d'introduire la réponse dans son entièreté par la frappe successive de tous les caractères constituant la réponse.

Pour faire son choix, il peut frapper le numéro précédent la réponse choisie ou la lettre placée entre les caractères (et). On passe alors directement à l'étape suivante.

Une autre possibilité consiste à se déplacer avec les touches 'flèches bas ' et 'flèche haut ' pour passer d'une réponse à l'autre jusqu'au moment où il se trouve sur la réponse souhaitée. La réponse courante est celle qui est précédée par le caractère '*' et qui apparaît après le libellé ' Votre choix : ' au bas de l'écran. Lorsqu'on se déplace d'une réponse à l'autre, la réponse choisie apparaît à la place de la précédente. L'utilisateur confirme son choix en utilisant la touche 'ENTER'.

La diversité des choix proposés doit conduire à une utilisation plus rapide du programme dans le cas d'un utilisateur averti.

9.2. CONSTRUCTION DU ROBOT.

Avant de commencer la simulation, il est obligatoire de construire le robot dont l'on désire simuler le comportement. Pour ce faire, il est possible soit de construire le robot pièce par pièce, soit de faire appel à un robot existant. Cela correspond à l'alternative résultant de l'écran 1.1.

Le premier point va d'abord exposer la lecture d'un robot existant tandis que le deuxième présente la création d'un nouveau robot.

9.2.1 LECTURE D'UN ROBOT EXISTANT.

Lors de sessions précédentes, l'utilisateur a eu l'occasion de mémoriser les robots dont il a simulé le comportement. Il lui est dès lors possible de travailler à nouveau avec un de ces robots.

Pour charger un robot, l'utilisateur dispose de deux moyens; Soit il demande tous les robots mémorisés à partir d'un nom d'utilisateur et choisit parmi tous les noms de robot proposés; Soit il demande un robot bien précis en donnant

le chemin complet pour y accéder. Le choix est fait par la sélection d'une des deux alternatives présentées à l'écran 2.1 .

S'il choisit la première option, l'utlisateur doit alors donner un nom d'utilisateur. Lors du sauvetage du robot, celui-ci est identifié en donnant, en outre, le nom de l'utilisateur qui sera considéré comme étant le 'propriétaire' du robot . Ce nom est saisi en utilisant l'écran 2.2 .

Après avoir saisi et confirmé le nom d'utilisateur, la validation de ce nom a pour but de vérifier que le nom d'utilisateur correspond bien à un nom connu du programme. Si la validation ne s'est pas déroulée correctement, on a l'affichage de l'écran 2.3 .

Si la validation s'est déroulée correctement, elle est suivie de l'affichage de tous les noms des robots disponibles sous le nom d'utilisateur saisi. Le choix s'effectue alors en se déplaçant d'un nom de robot à l'autre grâce aux touches 'flèche bas' et 'flèche haut'. Lorsque l'on arrive au sommet ou au bas de l'écran et si la liste des noms ne peut pas être mis sur la place disponible à l'écran, il y a un déplacement vers le haut ou vers le bas de la liste des noms. Le robot courant est précédé du caractère '*' comme on voit sur l'écran 2.4 .

L'utilisateur peut disposer de l'option AUTRE qui lui permet d'accéder à un robot directement. Pour ce faire, il faut donner le nom du robot. Ce nom est celui d'un robot qui n'est pas visible directement dans la partie de la liste affichée sur l'écran mais qui est disponible pour cet utilisateur. Cela permet de diminuer le temps de parcours de la liste des robots lorsque celle ci est longue dans la mesure où le nom du robot recherché est connu puisque le robot souhaité est accédé directement.

Si l'utilisateur choisit la deuxième alternative, il doit alors spécifier de façon précise tout le chemin à suivre dans les sous-directories pour accéder à ce robot. Pour accéder directement à ce robot, il est nécessaire de donner le nom de l'utilisateur sous le quel le robot cherché a été enregistré, le nom du robot même ainsi que la directory (ou suite de sous-directories) où le robot a été mémorisé. La saisie de ces trois champs est faite par l'écran 2.5 .

Si le robot n'est pas accessible selon le chemin désigné, l'utilisateur en est averti par l'écran 2.6 et il a alors la possibilité de spécifier un nouvel accès à un robot.

Quelque soit le moyen utilisé pour accéder à un robot mémorisé et dès l'accès réalisé, le robot est visualisé de manière similaire à celui présenté à l'écran 2.7.

L'utilisateur se trouve alors devant deux alternatives avant de continuer sa session:

La première consiste à accepter ce robot comme étant celui souhaité. L'utilisateur accepte alors le robot en frappant le caractère 'E' pour enregistrement et on passe à l'étape suivante. Dans ce cas, l'étape suivante offre la possibilité de modifier la structure actuelle du robot (Ce point sera décrit dans la partie Création du robot).

La deuxième possibilité consiste à refuser le robot sélectionné parce qu'il ne correspond pas à celui cherché. Dans ce cas, la frappe du caractère 'A' provoque l'abandon et le retour à l'étape précédente dans laquelle existe le choix d'utilisation d'un robot existant ou sa création.

Si, lors du sauvetage du robot désiré, on a mémorisé le fichier de trajectoire, celui-ci est chargé dans le disque virtuel. Après avoir choisi le robot qui va faire l'objet d'une nouvelle simulation, l'utilisateur doit indiquer s'il désire garder et afficher la trajectoire suivie lors de la dernière simulation exécutée avec ce robot. S'il désire la garder, celle suivie lors de la simulation courante est ajoutée à la suite de celle existante tandis que dans l'autre cas, celle existante est effacée

9.2.2. CREATION D'UN NOUVEAU ROBOT

Lorsqu'il n'existe pas de robot mémorisé ou que ceux mémorisés ne correspondent pas à ce que désire l'utilisateur, celui-ci peut en construire un, pièce par pièce.

Rappelons que le robot est une suite de bras numérotés séquentiellement à partir de 1 et attachés les uns aux autres à l'aide d'un objet de liaison. Chaque bras est défini à partir de ses caractéristiques qui sont le numéro de sa position dans la suite des bras, les valeurs affectées aux angles A et B et sa dimension. La suite de bras est terminée par un bras particulier appelé organe actif. La présence de celui ci est obligatoire et ses caractéristiques sont fixées arbitrairement.

Lors de la création d'un robot, il est nécessaire d'ajouter un à un les différents bras constituants le robot. Chaque bras pourra avoir ses caractéristiques modifiées ou être supprimé au cours de la création du robot.

L'utilisateur a la possibilité d'apporter les changements présentés dans l'écran 3.1 à la structure du robot. Les possibilités offertes sont les suivantes:

- ajouter un bras à l'ensemble des bras déjà existants
- modifier les caractéristiques d'un bras
- supprimer un bras jugé excédentaire

Il est à remarquer que, lorsque le robot n'a pas encore de bras, la seule possibilité de changer la structure est l'ajout d'un premier bras. Il est en effet exclu de vouloir faire d'autres changements à la structure d'un robot qui n'a pas encore d'éléments.

9.2.2.1. AJOUT D'UN BRAS

Lorsque l'utilisateur désire ajouter un bras, il lui faut décrire les propriétés que ce bras doit nécessairement avoir. En effet, pour définir un bras et en particulier sa position dans les référentiels actif et fixe, il faut connaître, en outre, les valeurs des angles A et B et la dimension et le type de liaison avec le bras précédent. La signification de ces données saisies est présentée dans le chapitre 3.

La première donnée devant être saisie est le numéro du bras. Ce numéro représente la position courante que doit prendre le bras considéré dans la suite des bras successifs qui forment le robot dans son état actuel. La numérotation doit être consécutive, elle doit débuter à 1 et il n'est pas possible de laisser un trou dans la numérotation des bras.

Puisque, dans la réalité, il est très rare de trouver des robots ayant plus de 5 bras, la simulation peut ainsi être limité à des robots ayant un maximum de 5 bras sans restreindre la généralité du système. La saisie du numéro du bras est assurée par l'écran 3.2 .

L'organe actif est un bras ayant un rôle particulier et des caractéristiques prenant des valeurs particulières. Il est identifié lors de la création par le numéro 0. D'autre part, ses caractéristiques (valeur des angles A et B et dimension) sont fixées et ne peuvent pas être modifiées.

L'organe actif est obligatoire, il doit être unique et ne peut placé que lorsqu'un autre bras a déjà été défini. Puisqu'il est obligatoire, il n'est pas possible de simuler le comportement d'un robot qui n'aurait pas d'organe actif. De même, il serait impensable de vouloir supprimer l'organe actif et placer un bras à la suite de cet organe actif serait insensé. Le rôle et la description de l'organe actif sont plus complètement présentés dans le chapitre 3.

Lorsque le numéro est connu et a été validé et dans la mesure où ce numéro ne correspond pas à celui de l'organe actif, il est nécessaire de saisir les caractéristiques du bras. Il s'agit de l'introduction d'une suite de 3 champs représentant la valeur des angles A et B et la dimension de l'axe directeur du bras. Les contraintes liées à ces valeurs sont uniquement en rapport avec leur caractère numérique et le respect des bornes affichées à l'écran. La saisie de ces champs se fait dans l'écran 3.3 .

Après avoir saisi correctement ces données et les avoir validées, il s'agit de sélectionner le type de liaison (rotule ou charnière) que l'on place entre le bras traité et le bras précédent ou la surface d'appui s'il s'agit du premier bras. Le choix de ce type de liaison se fait par sélection d'une des réponses présentées par

l'écran 3.4 . Les mouvements permis en fonction de chaque type de liaison sont définis dans le chapitre 3.

En fonction du type de liaison choisi, il faut pouvoir déterminer les contraintes de modification des valeurs des angles liés au type de liaison choisi. Ces contraintes se rapportent aux angles qui peuvent subir des variations en fonction du type de liaison choisi. Cela s'effectue par la sélection de la contrainte souhaitée dans l'écran 3.5 si le type de liaison choisi est une rotule et dans l'écran 3.6 si le type de liaison choisi est une charnière.

S'il s'agit du bras placé en première position, il est nécessaire de savoir si ce bras (et tous les autres bras à la suite) peut subir des translations le long des axes du référentiel actif et le long de quel(s) axe(s) ces translations sont possibles. Cela se fait également par la sélection de la réponse souhaitée dans l'écran 3.7 .

A noter que lorsque le numéro du bras saisi est 0, il ne faut rien saisir et l'utilisateur passe directement à la visualisation.

Après toutes ces saisies, on passe à la visualisation du robot dans sa structure actuelle.

9.2.2.2. MODIFICATION DES CARACTERISTIQUES D'UN BRAS

Lorsque l'utilisateur désire apporter des modifications aux caractéristiques d'un bras particulier, il est d'abord nécessaire d'identifier le bras concerné. L'utilisateur doit donc donner le numéro du bras dont il désire modifié les caractéristiques. Ce numéro doit être validé puisque il n'est pas possible de changer les caractéristiques d'un bras inexistant ni celles de l'organe actif.

Après avoir spécifié le bras concerné, il s'agit de saisir les nouvelles valeurs des caractéristiques (celles des angles A et B et celle de la dimension) qui sont affectées au bras.

Ensuite, il s'agit de définir le type de liaison qui va être placé.

De même, s'il s'agit de modifier le premier bras, il est nécessaire de déterminer le long de quel(s) axe(s) des translations sont permises.

La saisie des nouvelles valeurs des caractéristiques se fait de la même manière que celle faite dans le cas de l'ajout d'un bras et on a recours aux mêmes écrans.

Toutes ces saisies sont suivies de la visualisation de la structure actuelle du robot.

9.2.2.3. SUPPRESSION D'UN BRAS

Il n'est pas impensable que l'utilisateur se rende compte à un certain moment qu'un bras donné du robot est inutil ou impossible dans la réalité même s'il a été placé lors de la création du robot. Cette situation est possible eu égard au fait que le programme ne fait aucune interprétation sémantique des données saisies.

Il reste cependant la possibilité de supprimer ce bras. Il s'agit, dans ce cas, de saisir le numéro du bras qui doit être supprimé et de visualiser la structure actuelle du robot.

Il faut remarquer qu'il n'est cependant jamais possible de supprimer l'organe actif. En effet, puisque celui-ci est obligatoire, il serait impensable de le supprimer après l'avoir ajouté précédemment et ensuite devoir obligatoirement l'ajouter de nouveau.

9.2. LA VISUALISATION DE LA STRUCTURE DU ROBOT

Après chaque changement dans la structure du robot en cours de création, le robot est visualisé dans la structure résultant de la dernière modification. Cette visualisation est semblable à celle présentée dans l'écran 2.7 .

Il est alors possible de juger de l'opportunité du changement en fonction des besoins de la simulation.

Pour continuer la session, le programme propose deux alternatives.

Si la structure correspond aux désidérata de l'utilisateur et si celui ci désire conserver cette structure, il l'enregistre en utilisant le caractère 'E'. Il fait alors retours à l'étape où il peut décider de changer la structure du robot.

La structure peut ne pas correspondre à ce qu'il souhaite. Cela peut signifier qu'il a ajouté un bras alors qu'il ne le fallait pas ou qu'il l'a placé à un mauvais endroit ou encore qu'il a modifié les caractéristiques d'un bras alors que cela n'était pas nécessaire ou obligatoire ou enfin qu'il a supprimé un bras alors que l'il ne devait pas.

Dans le cas où la structure ne correspond pas aux souhaits de l'utilisateur, celui-ci a la possibilité de revenir à la dernière structure. Cela signifie que l'on en revient à la structure précédent le dernier changement (le bras ajouté est retiré, le bras modifié se voit réaffecté ses caractéristiques précédentes, le bras supprimé est replacé). Pour ce faire, l'utilisateur a à sa disposition la touche 'A' pour abandon. Cela a pour effet de reconstituer le robot dans la structure précédent le dernier changement. Cette possibilité offerte a pour but, en fait, de faire gagner du

temps à l'utilisateur qui ne doit pas, de cette manière, apporter les changement opposés à la structure du robot

Il y a alors la visualisation du robot dans cette structure précédente et on passe à l'étape suivante en enregistrant cette structure (touche 'E'). L'étape suivante est celle où l'utilisateur peut décider de changer la structure du robot.

9.3. DEPLACEMENT DU POINT FIXE

Dès qu'il en a terminé d'apporter des changements à la structure du robot et que celui ci représente parfaitement le robot existant, l'utilisateur décide de continuer et passe à l'étape suivante. Celle ci lui offre la possibilité de déplacer le point de liaison du robot avec la surface d'appui. Le robot utilisé peut avoir une structure telle qu'il dépasse la partie de l'écran qui lui est réservée. Il est alors préférable de pouvoir déplacer le robot et de le placer à un autre endroit sur l'écran. De cette manière, la visualisation du robot dont on fait des simulations est meilleure. Pour cela, il suffit de répondre de façon adéquate à la question de l'écran 4.1 .

Ce point de liaison peut être déplacé en tout endroit sur la partie de l'écran disponible pour le programme.

Ces déplacements sont possibles en utilisant les touches suivantes :
'flèche haut' pour déplacer le point fixe verticalement vers le haut.
'flèche bas' pour déplacer le point fixe verticalement vers le bas.
'flèche droite' pour déplacer le point fixe horizontalement vers la droite.
'flèche gauche' pour déplacer le point fixe horizontalement vers la gauche.
Cela est visualisé par l'écran 4.2 .

Lorsque le robot est placé à l'endroit souhaité sur l'écran, on accède à l'écran suivant qui termine la lecture-création d'un robot en affichant le robot et en le plaçant à l'endroit souhaité.

L'utilisateur fait alors exécuter des mouvements au robot en lui envoyant des commandes selon la marche à suivre décrite dans le chapitre 5.

9.4. SAUVETAGE DU ROBOT

Un utilisateur peut souhaiter mémoriser un robot dans le but de pouvoir aller le rechercher lors d'une session de travail ultérieure et faire de nouvelles simulations. Pour ce faire, il lui faut répondre de la façon indiquée à la question posée dans l'écran 5.1 . Le robot est mémorisé dans sa position de base soit celle déterminée par la position que chacun des bras le constituant avait lors de la

construction. La position qu'il avait lors de la dernière simulation n'est pas mémorisée et seule sa position initiale est retenue.

Pour assurer le sauvetage du robot, il est nécessaire de pouvoir l'identifier. Cette identification est réalisée en donnant le nom de l'utilisateur ('propriétaire') pour lequel le robot a été créé et le nom du robot lui-même. Puisque le système peut être utilisé par plusieurs utilisateurs différents, il est nécessaire de connaître l'utilisateur particulier pour lequel le robot a été construit. Cela doit permettre de retrouver tous les robots de chaque utilisateur. Le nom du robot est nécessaire pour identifier chacun des robots pour un utilisateur donné. L'écran 5.2 est prévu pour la saisie du nom de l'utilisateur et de celui du robot.

Cette identification est nécessaire afin de pouvoir reconnaître tous les robots de chacun des utilisateurs répertoriés par le système. En effet, chaque nom d'utilisateur est mémorisé et le relevé de tous les robots associés à chacun des utilisateurs est fait dans un fichier prévu à cet effet. Il en est de même pour le chemin pour y accéder. Cela facilite la lecture des robots mais également assure l'unicité de chaque robot pour un utilisateur.

Après avoir identifié le robot qui est l'objet de la mémorisation, il faut spécifier la directory de sauvetage dans laquelle se fait le sauvetage. Il est en effet préférable de sauver les robots à des endroits connus et que l'utilisateur se rappèlera facilement. Cela facilite la lecture lors d'une session de travail ultérieure. Cependant, si l'utilisateur désire avoir recours à la directory de sauvetage par défaut qui est spécifiée dans le fichier de configuration du programme, il utilisera la touche F1 pour confirmer qu'il fait appel à cette directory par défaut.

D'autre part, l'utilisateur peut décider de mémoriser le robot dans une directory particulière. Il doit alors utiliser la touche F3 dont l'activation provoque l'affichage de l'écran 5.4 dans lequel on demande le nom complet de la directory ou suite de sous-directories qu'il a choisi pour le sauvetage. Dans ce cas, il y a une validation du chemin afin d'éviter la saisie de directories réservées pour le système. Deux directories sont réservées pour le système, à savoir DATA et UTILIS. Le chemin spécifié ne doit pas nécessairement avoir été créé au préalable, le système se chargeant de le créer si cela s'avère nécessaire.

Lorsque le nom du robot est redondant avec celui d'un autre robot ayant déjà été mémorisé pour cet utilisateur particulier, il faut alors soit écraser le contenu du robot initial soit abandonner le sauvetage. L'utilisateur doit choisir ce qu'il désire faire en répondant à la question de l'écran 5.3. S'il choisit l'écrasement, c'est le robot que l'on veut sauver qui est mémorisé à la place de l'autre. Dans le cas contraire, on passe à l'étape suivante qui correspond à la sortie du système.

Quand le sauvetage est terminé, on passe à l'étape suivante correspondant également à la sortie du système.

Avant et après la saisie de chacun des champs lors de l'étape de sauvetage, il est possible de stopper le sauvetage. Il suffit pour cela d'utliser la touche F2 prevue à cet effet.

Il faut remarquer qu'il n'est pas possible de détruire un robot à partir du système.

Si, lors de la configuration du système, l'utilisateur a exprimé le souhait de visualiser la trajectoire suivie par l'organe actif et de garder cette trajectoire dans un fichier, ce fichier est mémorisé lors du sauvetage du robot. Le sauvetage de ce fichier se fait au même endroit que le robot et ce fichier a pour nom, le nom du robot suivi de l'extention .TRA .

Afin de pouvoir faire des vérifications, il peut être intéressant de connaître le chemin complet pour accéder aux fichiers contenant le robot et la trajectoire suivie si celle ci est mémorisée.

Le chemin complet pour accéder à l'endroit où sont mémorisés le fichier contenant le robot et celui contenant la trajectoire est déterminé de la manière suivante:

Il y a d'abord le nom de la directory de sauvetage. Il s'agit soit de la directory par défaut spécifiée lors du programme de configuration, soit de la directory saisie à partir de l'option F3 de l'écran 5.4.

Dans cette directory de sauvetage, on trouve une directory dont le nom correspond à celui de l'utilisateur sous lequel a été mémorisé le robot.

Le fichier du robot et celui de la trajectoire sont mémorisés dans cette sous-directory

9.5.CONFIGURATION DU SYSTEME

Un programme aussi complexe que RESIDENT doit pouvoir s'exécuter dans un grand nombre d'environnements différents et sur une grande variété de configuration informatique. Ces différences peuvent provenir des caractéristiques du matériel disponible mais également de l'utilisateur et de son environnement (objectifs ergonomiques, langue utilisée, ...) .

Dès lors, en vue d'une utilisation correcte, il est nécessaire d'avoir la définition de l'environnement dans lequel le programme va s'exécuter.

L'environnement d'un programme comporte des éléments très diversifiés. Cela va du matériel hardware disponible aux contraintes imposées en vue d'avoir un programme relativement convivial et des désidérata de l'utilisateur.

Les caractéristiques hardware d'un micro-ordinateur peuvent être différentes. Pour ce programme, elles sont relatives au nombre de disques durs et au nombre de lecteurs de disquette mais également à la carte graphique dont on dispose. Il s'agit, dans ce cas, d'éléments jouant un rôle en vue d'assurer l'exécution correcte du programme.

De plus, en vue d'atteindre un aspect ergonomique suffisant, il peut être intéressant de permettre à l'utilisateur de disposer de certaines options qui permettent d'avoir une utilisation conviviale du programme. C'est pourquoi il est permis à l'utilisateur de faire le choix de la langue ou de la couleur qu'il souhaite utiliser mais également les directories choisies par défaut lors du sauvetage d'un robot ainsi que le traitement de la trajectoire qu'il désire réaliser. D'autre part, il faut également prévoir la possibilité de poser des choix internes à la robotique.

Le fichier de configuration contient tous ces renseignements et son contenu doit être accessible à tout moment par le programme puisque les informations contenues sont régulièrement consultées afin que le programme s'adapte aux caractéristiques de l'environnement et aux désirs de l'utilisateur.

Pour la détermination de chacun des champs du fichier, on passe dans la suite des écrans du programme de configuration et on sélectionne à chaque étape la valeur souhaitée pour chaque champs

9..5.1.CONFIGURATION HARDWARE

Afin d'avoir des performances valables lors de l'affichage des écrans, il a été décidé d'avoir recours à la technique du disque virtuel dans lequel sont chargés tous les fichiers de type texte auxquels il est fait appel lors des dialogues avec l'utilisateur. Grace à cette technique du disque virtuel, l'accès à ces fichiers se fait de manière beaucoup plus rapide puisqu'il peut être assimilé à un accès à la mémoire interne du système.

Le programme doit dès lors pouvoir connaître l'identification du drive correspondant au disque virtuel. Pour ce faire, il est nécessaire de connaître le nombre de lecteur de disquettes et le nombre de disques durs dont on dispose pour pouvoir détecter l'identificateur du drive correspondant au disque virtuel. Cet identificateur est la lettre (C,D, ...) associée au drive 'disque virtuel'.

La configuration est limitée à un maximum de 2 lecteurs de disquettes et 2 disques durs. Cela correspond à la configuration la plus puissante de ce point

de vue dont on dispose pour faire exécuter le programme. En effet, il est rare de disposer de configurations plus développées. La configuration minimale est d'avoir un seul lecteur de disquette.

9.5.2.CARTE GRAPHIQUE

La carte graphique est une autre caractéristique de l'ordinateur sur lequel on exécute le programme. Puisque le programme de simulation doit visualiser des robots sous une forme graphique, il est nécessaire de connaître la carte dont l'ordinateur est muni afin d'assurer une bonne représentation à l'écran. Cette carte graphique peut être de différents types (VGA, CGA, EGA, HERCULES).

La connaissance de la carte graphique disponible permet de déterminer la façon (nombre de points par unité, grandeur des côtés, ...) dont on dessinera les bras constituants le robot.

La détection de la carte se fait de manière automatique car il est nécesssaire d'avoir une correspondance parfaite entre la carte choisie et celle effectivement placée. Pour prévenir toute erreur dans la détection de la carte, celle ci est réalisée par le programme lui-même lors de l'initialisation du mode graphique.

9.5.3.CHOIX DE LA LANGUE

Le programme est destiné à des utilisateurs pouvant venir de pays différents ou désireux de travailler dans différentes langues. Dans ce cas, il peut être intéressant que les messages et contenus des écrans soient écrits dans plusieurs langues. De cette façon, l'utilisateur a la possibilité de choisir la langue de travail qu'il désire utiliser.

Il faut remarquer que cette possibilité d'avoir des messages en plusieurs langues n'est pas valable pour les commandes de mouvement du robot puisque ces commandes doivent respecter un format très stricte et être syntaxiquement et sémantiquement correctes.

9.5.4.CHOIX DE LA COULEUR

L'utilisateur a la possibilité de choisir la couleur dans laquelle les bras du robot sont affichés à l'écran.

Cela correspond à un objectif ergonomique visant à avoir une visualisation la plus agréable possible pour chacun des utilisateurs.

9.5.5.SOUS-DIRECTORY PAR DEFAUT

Un utilisateur peut vouloir sauver ses robots sur des mémoires permanentes en vue de pouvoir aller les rechercher lors d'une session ultérieure.

Lors du sauvetage d'un robot dans un fichier, il faut définir le drive et éventuellement le chemin d'accès à l'emplacement où on placera le fichier.

L'utilisateur est entièrement libre de placer ses robots à l'endroit qu'il désire. Cependant, il n'est pas insensé d'imaginer qu'un utilisateur souhaite regrouper ses robots mémorisés et qu'il désire les sauver dans une même directory.

C'est pourquoi il lui est donné la possibilité de définir une directory particulière (directory par défaut) dans laquelle il placera systématiquement les robots qu'il désire sauver sauf indication contraire lors du sauvetage lors de la réalisation du sauvetage.

Une directory par défaut est proposée à l'utilisateur. Cette directory est créée et intégrée au système. L'utilisateur ne doit donc pas la créer et il peut en choisir une autre que celle déjà créée. Le programme se charge de la créer si cela est nécessaire.

De même, lors de l'opération de sauvetage d'un robot, l'utilisateur a la possibilité de sauver le fichier contenant le robot dans une directory particulière, différente de celle définie par défaut.

9.5.6. SAUVETAGE DE LA TRAJECTOIRE

Un utilisateur peut trouver intéressant de pouvoir garder la trajectoire suivie par l'organe actif du robot.

Cette trajectoire peut être mémorisée dans un fichier qui est placé dans le disque virtuel et apparaîtra à l'écran lors de chacune des visualisations des robots. Elle est représentée par la liaison par des segments de droites entre les différents points par lesquels est passé l'organe actif lors des simulations de mouvements.

Cette trajectoire peut également être observée selon différents points de vue de la même manière qu'il est possible de voir le robot sous différents points de vue.

9.5.7. CHOIX RELATIF AUX VARIATIONS DES ANGLES A ET B

Comme cela a été présenté dans le chapitre 3., les bras suivant un bras qui subit une variation d'un de ses angles A et B peuvent se comporter de 2 manières différentes.

Dans le cas de la première alternative, les bras suivants gardent la même position relative par rapport au bras traité tandis que pour la deuxième, ils changent de position relative mais gardent la même position absolue puisque la valeur de leurs angles A et B ne sont pas modifiées.

L'utilisateur a la possibilité de simuler le comportement de robots réagissant des 2 manières. Il lui suffit d'indiquer son choix dans le fichier de configuration.

1-1-1980

FNDP-NAMUR ENIDH-LISBOA

SimRob

Realisateurs : Edgard CONRARD
David DARTE

Promoteurs : Silvestre ANTUNES
Jorge BARRETO

(C)ontinuer Aide : F10

ECRAN 1.1

SimRob

Desirez-vous utiliser un robot existant [O/N] :

Ecran initial :home Aide : F10 Votre choix :

ECRAN 1.1

SimRob

*(1) Robot pour un utilisateur particulier

(2) Robot dans une directory particuliere

Ecran initial :home Ecran précédent :pgup Aide :F10

Votre choix : (1) Robot pour un utilisateur particulier

ECRAN 2.1

SIMROB

NOM DE L'UTILISATEUR : *

Ecran initial :home Ecran précédent :pgup Aide :F10

ECRAN 2.2

ECRAN 2.4

S I M R O B

nono

(A)utre

Ecran precedent : Pgup Ecran initial: Home Choix : Aide :F10

SimRob

*NOM DE L'UTILISATEUR :

NOM DU ROBOT :

NOM DE LA DIRECTORY :

CONFIRMER :F1

ABANDONNER :F2

Ecran initial :home Ecran précédent :pgup Aide :F10

ECRAN 2.5

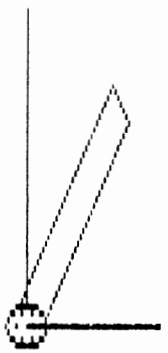
SIMROB

LE NOM DE L'UTILISATEUR EST INCONNU DU SYSTEME .

VOULEZ-VOUS INTRODUIRE UN NOUVEAU NOM [O/N]:

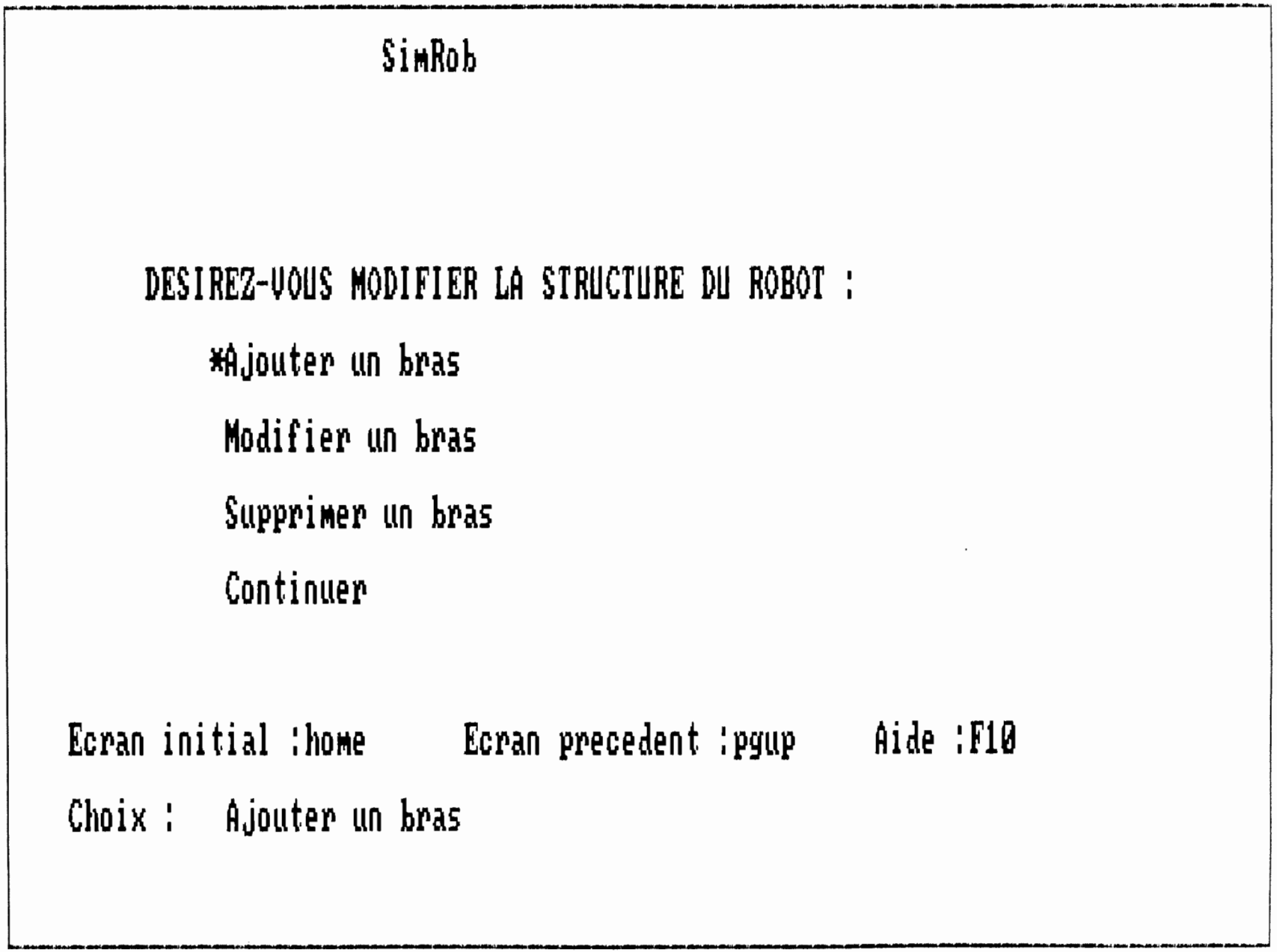
Ecran initial :home Ecran precedent :pgug Choix : Aide :F10

ECRAN 2.6



(E)nregistrement (A)bandon Aide :F10

ECRAN 2.7


SimRob
DESIREZ-VOUS MODIFIER LA STRUCTURE DU ROBOT :
*Ajouter un bras
Modifier un bras
Supprimer un bras
Continuer
Ecran initial :home
Ecran precedent :pgup
Aide :F10
Choix : Ajouter un bras

ECRAN 3.1

SimRob

Veuiller introduire le numero du bras à traiter(0 pour l'organe actif):

Votre choix : *

Ecran initial :home Ecran précédent :pgup Aide :F10

ECRAN 3.2

SimRob

Veuiller introduire les renseignements relatifs au bras1

*Angle A (0<=A<=360) :

Angle B (0<=B<=360) :

Dimension (1..5) :

Ecran initial :home Ecran précédent :pgup Aide :F10

Enregistrement des donnees :E

ECRAN 3.4

SimRob

Quelle est la liaison du bras1 avec le bras precedent :

*ROTULE

CHARNIERE

Ecran initial :home Ecran précédent :pgup Aide :F10

Votre choix : ROTULE

ECRAN 3.4

SimRob

Quels sont les mouvements permis pour la rotule :

*(1) Variation de l'angle A uniquement

(2) Variation de l'angle B uniquement

(3) Variation des angles A et B

Ecran initial :home Ecran précédent :pgup Aide :F10

Votre choix : (1) Variation de l'angle A uniquement

ECRAN 3.5

SimRob

Quelles sont les mouvements premis pour la charniere :

*(1) Variation de l'angle A

(2) Variation de l'angle B

Ecran initiale :home Ecran précédent :pgup Aide :F10

Votre choix : (1) Variation de l'angle A

ECRAN 3.6

SimRob

Quels sont les axes le long desquels sont permis des translations :

*(0) Aucun axe

(1) L'axe X uniquement

(2) L'axe Y uniquement

(3) L'axe Z uniquement

(4) Les axes X et Y

(5) Les axes X et Z

(6) Les axes Y et Z

(7) Les axes X,Y et Z

Ecran initial :home Ecran précédent :pgup Aide :F10

Votre choix : (0) Aucun axe

ECRAN 3.7

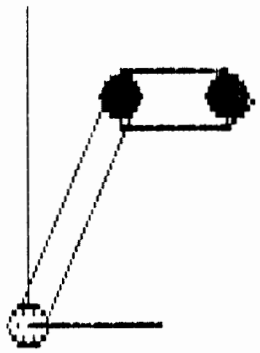
SimRob

Desirer vous deplacer l'origine du robot [O/N] ?

Votre choix :

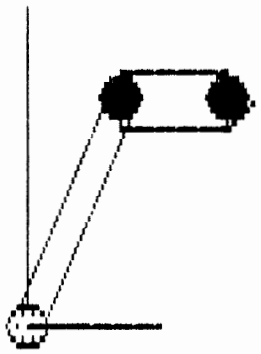
Ecran initial :home Ecran précédent :pgup Aide :F10

ECRAN 4.1



Monter :fleche haut Descendre :fleche bas
Gauche :fleche gauche Droite :fleche droite
Terminer :enter Aide :F10

ECRAN 4.2



VOICI VOTRE ECRAN DE TRAVAIL :
Ecran initial :home Ecran précédent : pgup
Continuer :enter Aide :F10

ECRAN 4.3

SimRob

Desirer vous sauver le robot actuel [O/N] :

Votre choix :

Aide :F10

ECRAN 5.1

SimRob

*NOM DE L'UTILISATEUR :

NOM DU ROBOT :

CONFIRMER : F1

ABANDONNER : F2

AUTRE DIRECTORY : F3

Aide : F10

ECRAN 5.2

SimRob

NOM DU DIRECTORY : *

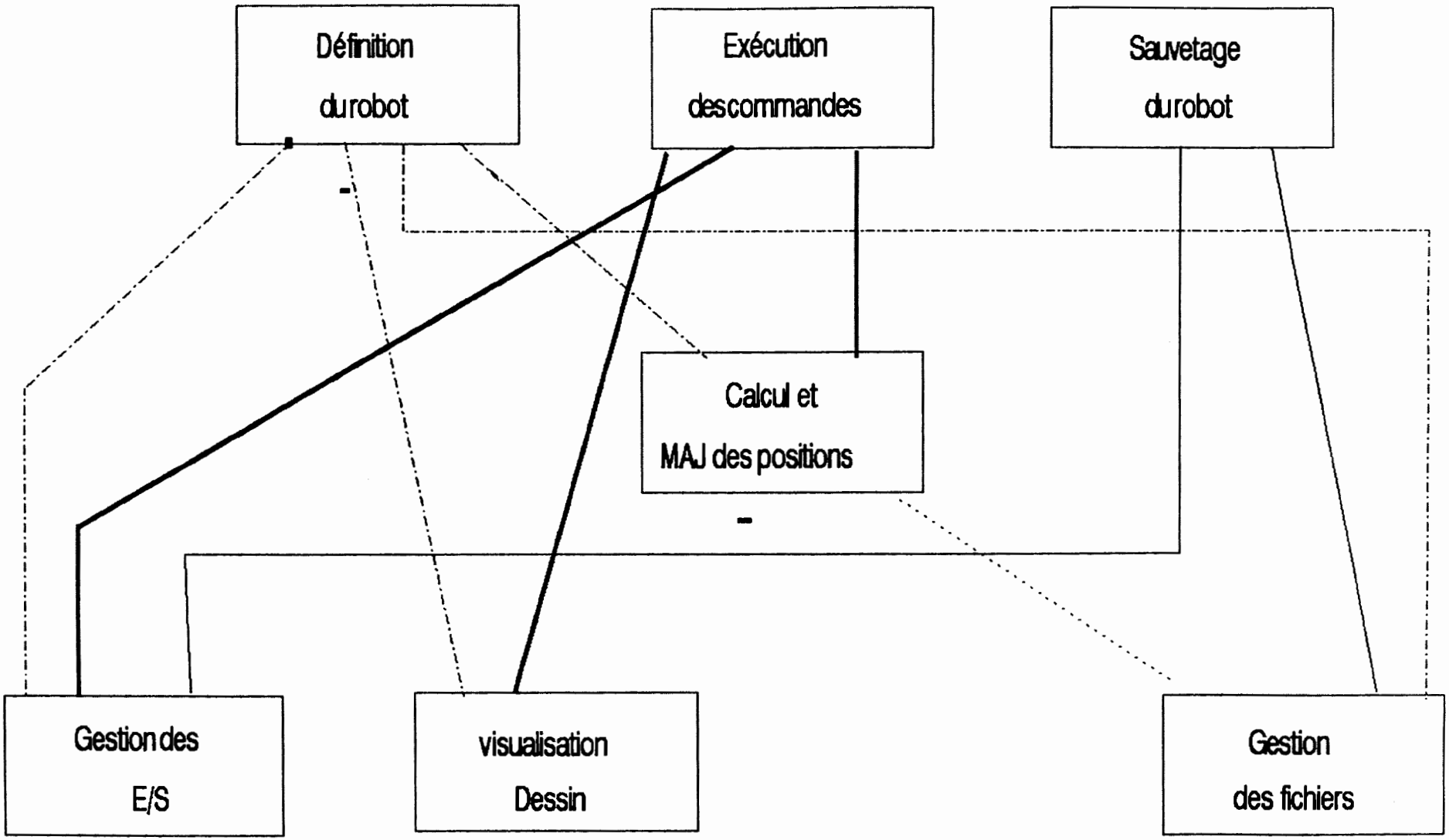
Ecran precedent : Pgup Aide :F10

ECRAN 5.4

ANNEXES

ANNEXE 1

Architecture logique.



Architecture logique : relation utilise

ANNEXE 2

Architecture physique

ARCHITECTURE PHYSIQUE.

NIVEAU 0

Résident appelle:

INITSIMR
LECTURE
LIRE_COMMANDE
INTERPRE
ROTAT1
ROTAT2
ROTAT3
ROTAT4
ECRANTRA
IMPRIMTRA
SAUVE_POS_INTERM
RETOUR_POS_INTERM
RETOUR_INITIAL
RETOUR_INTERM
SAUVETAG_ROBOT

NIVEAU 1

1.1 Lecture appelle:

ECRAN1/ECRAN2/ECRAN90/ECRAN9
ECRAN93/ECRAN3
ECRAN5/ECRAN10/ECRAN101/ECRAN102
/ECRAN111
ECRAN112/ECRAN6/ECRAN9/ECRAN12/
ECRAN40/ECRAN41/
ECRAN42/ECRAN43/ECRAN44/ECRAN7
/ECRAN8/ECRAN15
TRAIT_ORG_ACTIF
INITDONN
LECTURE_ROBOT
LIREAXE/LIREANGL/LIREPOS

1.2 Initsimr appelle:

INITECRAN
INITDIRECTORY

INITLANGUE
INITRAJ

1.3 Rotat1, Rotat2, Rotat3 et Rotat4 appellent :

COORDAXE
COORDFIXE
COORDORI
MAJTRAJAXE
DESSINAXES
DESSINBRAS

1.4 Ecrantra appelle :

DESSINBRAS
DESSINAXES

1.5 Imprimtra et Sauv_pos_interm appellent :

Néant

1.6 Retourinitial et retour_interm appellent :

DESSINBRAS
DESSINAXES

1.7 Sauver appelle :

LECTURE_CHEMIN
VERIFICATION_CHEMIN
ROBOT_EXIST
CREATION_CHEMIN
SOUS_DIRECTORY_OK
AJOUT_NOM_UTILIS
CREATION_FICH_ROBOT
COPIE_ROBOT_DIM
ECRANSV2/ECRANSV3

ANNEXE 3

Spécifications

UNIT COSSIN

PROCEDURE DETENDRE

E : Angl : valeur réelle d'un angle compris entre 0 et 360

E-S : ·

S : Résult : valeur réelle de l'angle Angl ramené au premier quadrant.

BUT : Evident

APPEL : ·

PROCEDURE SUP360

E : ·

E-S : Angl : valeur réelle d'un angle quelconque ramené à un angle compris entre 0 et 360 si la valeur de départ est supérieure à 360

S : ·

BUT : Evident

APPEL : ·

PROCEDURE COSTAB

E : Angl : valeur réelle représentant un angle quelconque

E-S : ·

S : Result : valeur réelle représentant le cosinus de l'angle Angl, quelque soit sa valeur

BUT : Evident

APPEL : ·

PROCEDURE SINTAB

E : Angl : valeur réelle représentant un angle quelconque

E-S : ·

S : Result : valeur réelle représentant le sinus de l'angle Angl, quelque soit sa valeur

BUT : Evident

APPEL : ·

PROCEDURE RECHERCHE

E : cs : valeur du cosinus d'un angle quelconque

E-S : ·

S : Angl : valeur comprise entre 0 et 360 de l'angle

dont le sinus est cs

BUT : Evident

APPEL : ·

PROCEDURE TABCOS

E : cs : valeur du cosinus d'un angle quelconque

E·S : ·

S : Angl : valeur comprise entre 0 et 360 de l'angle dont le cosinus est cs

BUT : Evident

APPEL :·

PROCEDURE TABSIN

E : sn : valeur du sinus d'un angle quelconque

E·S : ·

S : Angl : valeur comprise entre 0 et 360 de l'angle dont le cosinus est sn

BUT : Evident

APPEL :·

UNIT VARANGLE

PROCEDURE VARANGX

E : Angx : valeur représentant la variation de l'angle x qu'il y a entre l'axe X_0 du référentiel écran et l'axe X du référentiel actif du robot

E·S :

S :

BUT : Calculer les coordonnées des points unités du système d'axes du référentiel actif du robot dans le référentiel écran et mettre à jour la zone mémoire réservée pour mémoriser ces coordonnées

APPEL : COSSIN : COSTAB · SINTAB

TABLEAUX : TAXE_MEM

PROCEDURE VARANGY

E : Angy : valeur représentant la variation de l'angle y qu'il y a entre l'axe Y_0 du référentiel écran et l'axe Y du référentiel actif du robot

E·S :

S :

BUT : Calculer les coordonnées des points unités du système d'axes du référentiel actif du robot dans le référentiel écran et mettre à jour la zone mémoire réservée pour mémoriser ces coordonnées

APPEL : COSSIN : COSTAB · SINTAB

TABLEAUX : TAXE_MEM

PROCEDURE VARANGZ

E : Angz : valeur représentant la variation de l'angle z qu'il y a entre l'axe Z_0 du référentiel écran et l'axe X du référentiel actif du robot

E-S :

S :

BUT : Calculer les coordonnées des points unités du système d'axes du référentiel actif du robot dans le référentiel écran et mettre à jour la zone mémoire réservée pour mémoriser ces coordonnées

APPEL : COSSIN : COSTAB · SINTAB

TABLEAUX : TAXE_MEM

PROCEDURE CALRAC

E : Num : numéro du bras traité

E-S : ·

S : Rac : valeur réelle égale à la somme des carrés des coordonnées du point directeur du bras de numéro Num dans le référentiel actif de ce bras.

BUT : Calculer Rac à partir des valeurs correspondant au bras Num et mémorisées dans la zone mémoire stockant le robot.

APPEL : TABLEAUX : MEM_TROB

PROCEDURE CALV1V2V3

E : Num : numéro du bras traité

Rac : la valeur calculée par la procédure CALRAC

E-S : ·

S : V1 : valeur qui est le rapport entre la coordonnée x du point directeur du bras de numéro Num dans le référentiel actif de ce bras et la valeur Rac

V2 : valeur qui est le rapport entre la coordonnée y du point directeur du bras de numéro Num dans le référentiel actif de ce bras et la valeur Rac

V3 : valeur qui est le rapport entre la coordonnée z du point directeur du bras de numéro Num dans le référentiel actif de ce bras et la valeur Rac

BUT : Evident

APPEL : TABLEAUX : MEM_TROB

PROCEDURE CALCSK

E : Angl : valeur d'un angle quelconque compris entre 0 et 360

E-S : ·

S : Vcos : valeur du cosinus de l'angle Angl

Vsin : valeur du sinus de l'angle Angl
K = 1· Vcos

BUT : Evident

APPEL : COSSIN : COSTAB · SINTAB

PROCEDURE DETERMMAT1

E : K : valeur fournie par la procedure CALCSK
V1,V2,V3 : valeur fournie par la procedure CALV1V2V3

E·S :

S : MAT1 : matrice 3*3 calculée à partir des valeurs K,V1,V2,V3

BUT : Evident

APPEL : ·

PROCEDURE DETERMMAT2

E : Vcos, Vsin : valeur fournies par la procedure CALCSK

V1,V2,V3 : valeurs fournies par la procedure CALV1V2V3

E·S : ·

S : MAT2 : matrice 3*3 calculée à partir des valeurs Vcos, Vsin, V1, V2, V3

BUT : Evident

APPEL : ·

PROCEDURE DETERMMAT3

E : MAT1 : matrice 3*3 fournie par la procedure DETERMMAT1

MAT2 : matrice 3*3 fournie par la procedure DETERMMAT3

E·S :

S : MAT3 : matrice 3*3 résultant de la somme des 2 matrices MAT1 et MAT2

PROCEDURE FCKEY1

E : ch : caractère lu au terminal ou dans un fichier et représentant une variation d'un des angles qui existent entre le référentiel écran et le référentiel du robot.

vari : caractère indiquant l'axe concerné par la variation

pas : amplitude de la variation

E·S :

S :

BUT : Mettre à jour la zone mémoire stockant le tableau des axes en répercutant la variation lue et en calculant la nouvelle position du référentiel actif du robot

APPEL : TABLEAUX : MEM_TAXE · TAXE_MEM

PROCEDURE VARIAT1

E : bornex, borney :valeur réelle servant lors de la présentation à l'écran

pas : l'amplitude de la variation

E-S :

S : vari : caratère désignant l'axe concerné.

BUT : Repercuter les variation dans la zone mémoire contenant les coordonnées des axes

APPEL : VARANGLE : FCKEY1

PROCEDURE FCKEY2

E : ch : caractère indiquant l'angle qui subit une variation ainsi que le sens de la variation

num : le numéro du bras traité

pas : l'amplitude de la variation

E-S :

S :

BUT : Mettre à jour la zone mémoire contenant le robot en répercutant les variations sur l'angle désigné du bras dont le numéro est Num

APPEL : TABLEAUX : MEM_TROB · TROB_MEM

PROCEDURE VARIAT2

E : Angl : désignant l'angle A ou B que l'on veut faire varier

Pas : la grandeur de la variation

sig : le signe de la variation

Num : numéro du bras traité

E-S :

S :

BUT : répercuter les modifications désirées dans la zone mémoire contenant le robot

APPEL : VARANGLE : FCKEY2

PROCEDURE FCKEY3

E : ch : caractère indiquant l'axe le long duquel on désire faire une translation et le sens de la translation

Num : le numéro du bras traité

Pas : l'amplitude du mouvement

E-S :

S :

BUT : Mettre à jour la zone mémoire réservée pour mémoriser l'origine de chacun des bras du robot

APPEL : TABLEAUX : MEM_TORI · TORI_MEM

PROCEDURE VARIAT3

E : axe : caractère indiquant l'axe traité

sig : caractère indiquant le sens de la translation

pas : l'amplitude de la translation

E·S :

S :

BUT : Répercuter la translation dans la zone mémoire adéquate

APPEL : VARANGLE : FCKEY3

PROCEDURE VARIAT4

E : pas : l'amplitude du pivotage

sign : caractère indiquant le sens du pivotage

E·S :

S : Ang : la grandeur réelle du pivotage

BUT : Calculer la grandeur du pivotage d'un bras sur lui même à partir de l'amplitude et du sens

APPEL : ·

UNIT COORDONN

PROCEDURE COORDACTIF

E : num : le numéro du bras traité

E-S :

S :

BUT : Calculer les coordonnées dans le référentiel actif du du bras de numéro num des points relatifs au bras de ce numéro et mettre à jour la zone mémoire contenant le robot. Le calcul se fait à partir de la valeur des angles A et B

APPEL : COSSIN : COSTAB

TABLEAUX : TROB_MEM

PROCEDURE COORDROT

E : num : le numéro du bras traité

mat3 : une matrice 3*3 servant aux calculs des coordonnées dans le référentiel actif d'un bras

angle : l'angle de pivotage d'un bras sur lui même

BUT : Calculer les coordonnées dans le référentiel actif du bras num des points définissant ce bras lorsqu'il a subi un pivotage sur lui même d'un angle Angl. Ce calcul se fait à partir des anciènnes coordonnées et de la matrice mat3

APPEL : VARANGLE : CALRAC · CALV1V2V3 · CALCSK · DETERMMAT1 · DETERMMAT2 · DETERMMAT3

TABLEAUX : TROB_MEM

PROCEDURE CALCULA

E : num : le numéro du bras traité

E-S :

S :

BUT : Calculer la valeur de l'angle A associé au point identifié par le numéro num à partir des coordonnées dans le référentiel actif . Ces valeurs sont placées dans la zone mémoire contenant le robot. De même la valeur de l'angle A obtenue sera placé dans cette zone

APPEL : COSSIN : TABCOS ·TABSIN

TABLEAUX : TROB_MEM

PROCEDURE CALCULB

E : num : le numéro du bras traité

E-S :

S :

BUT : Calculer la valeur de l'angle B associé au point identifié par le numéro num à partir des coordonnées dans le référentiel actif . Ces valeurs sont placées dans la zone mémoire contenant le robot. De même la valeur de l'angle b obtenue sera placé dans cette zone

APPEL : COSSIN : TABCOS ·TABSIN

TABLEAUX : TROB_MEM

PROCEDURE VALEURAB

E : num : numéro du bras traité

E·S :

S :

BUT : Calculer les valeurs des angles A et B des 3 points définissant le bras de numéro num et mettre à jour la zone mémoire contenant le robot

APPEL : COORDONN : CALCULA · CALCULB

TABLEAUX : TROB_MEM · MEM_TROB

PROCEDURE COORDFIXE

E : num : le numéro du bras traité

E·S :

S :

BUT : calculer les coordonnées dans le référentiel écran de chacun des points définissant le bras de numéro num et mettre à jour la zone mémoire contenant le robot. Pour ces calculs, il est nécessaire d'avoir le contenu des zones mémoires stockant les origines de chaque bras et les axes.

APPEL : TABLEAUX : TROB_MEM · MEM_TROB · MEM_TAXE · MEM_TORI

PROCEDURE COORDORI

E :

E·S :

S :

BUT : Calculer les coordonnées dans le référentiel écran du point origine de chacun des bras du robot et mettre à jour la zone mémoire contenant ces coordonnées. Pour ce calcul, il est nécessaire de connaître le contenu de la zone mémoire stockant les axes.

APPEL : TABLEAUX : MEM_TORI · TORI_MEM · MEM_TAXE

PROCEDURE COORDAXE

E : vari : caractère indiquant l'axe du référentiel actif du robot dont l'angle avec l'axe correspondant dans le référentiel écran varie.

E·S :
S :
BUT : Calculer les coordonnées dans le référentiel écran des 3 points unités du système d'axes du référentiel du robot et mettre à jour la zone mémoire contenant ces axes.

APPEL : VARANGLE : VARANGX · VARANGY · VARANGZ

TABLEAUX : MEM_TAXE

PROCEDURE COORDTRAJ

E :

E·S : ligtraj : une ligne du fichier contenant l'ensemble des points par lesquels la trajectoire suivie par l'organe actif est passée lors de la simulation

S :

BUT : Calculer les coordonnées dans le référentiel écran du point mémorisé dans cette ligne du fichier à partir de ces coordonnées dans le référentiel du robot

APPEL : TABLEAUX : MEM_TAXE

PROCEDURE MAJTRAJ

E : donn : dataconfig

E·S :
S :
BUT : Ajouter dans le fichier contenant la trajectoire suivie par l'organe actif, les coordonnées dans le référentiel actif du robot du dernier points (c'est à dire l'extrémité du dernier bras)
APPEL : TABLEAUX : MEM_TORB · MEM_TORI

PROCEDURE MAJFTRAJAXE

E :
E·S :
S :
BUT : Calculer les coordonnées dans le référentiel écran de tous les points par lesquels est passé l'organe actif

APPEL : COORDONN : COORDTRAJ

UNIT OUTIL

PROCEDURE COPIEROB

E : couleur

coordx, coordy

hauteur

unite

bornex, borney

Ces données servent lors de la visualisation d'un robot à l'écran

donn : dataconfig

E-S :

S :

BUT : Visualiser le robot mémorisé dans la zone mémoire réservée à cet effet.

APPEL : DESSIN : DESSINBRAS · DESSINAXES

PROCEDURE INITDONN

E :

E-S : tch : tableau de strings servant à mémoriser temporairement un ensemble de données numériques saisies sous forme alphabétique.

S :

BUT : Initialiser le tableau tch en mettant tous ses éléments à blanc

APPEL :

PROCEDURE COPIE_TAB_MEM

E : t1 : tableau intermédiaire de type tabrob et servant à mémoriser temporairement tout ou une partie du robot.

num : le numéro du bras en cours de traitement
deb : l'indication de l'endroit de début de traitement

E-S :

S :

BUT : Copier le tableau t1 dans l'espace mémoire contenant le robot en commençant à la ligne désignée par deb ce à partir de la zone défini grâce à num

APPEL : TABLEAUX : TROB_MEM

PROCEDURE MEM_COPIETAB

E : num : le numéro du bras à partir duquel on doit commencer à copier

rer des éléments

E-S :

S : t1 : tableau de type tabrob pouvant contenir tout ou partie d'un robot.

BUT : Copier la partie de la partie commençant à la ligne fixée par num de la zone mémoire contenant le robot dans le tableau t1 à partir de la ligne calculée grâce deb

APPEL : TABLEAUX : MEM_TROB · TROB_MEM

PROCEDURE DETERMVAL

E : lig : un ensemble de caractère représentant un nombre

E-S :

S : résult : valeur numérique correspondant au string lig

BUT : Calculer la valeur numérique du string lig

APPEL :

PROCEDURE COPIEVAL

E : valeur : un nombre quelconque

num : le numéro de la ligne à complèter

ind : la position dans la ligne

E-S :

S : t : tableau de type tabrob pouvant contenir un robot

BUT : Copier la valeur Valeur dans la colonne pos de la ligne num du tableau t

APPEL :

PROCEDURE TRAITTCH

E : tch : tableau intermédiaire contenant un ensemble de valeurs réelles sous la forme de string

E-S :

S : trob1 : tableau temporaire de type tabrob pouvant contenir tout ou partie d'un robot

BUT : Traiter le tableau tch et construire simultanément le tableau trob1 à partir de son contenu, chaque élément de tch ayant une signification et une valeur particulière.

APPEL : OUTIL : COPIEVAL · DETERMVAL

PROCEDURE MAJA

E : tch : tableau intermédiaire contenant un ensemble de valeurs numériques sous la forme alphabétique

trob1 : tableau intermédiaire pouvant contenir un robot

E-S :

S :

BUT : Mettre à jour la partie mémoire où est stocké le robot lors de l'ajout d'un bras. L'ancien contenu de la zone est gardé pour le cas où il faudrait le retrouver.

APPEL : OUTIL : DETERMVAL · MEM_COPIEVAL · COPIE-VA_MEM · TRAITTCH

TABLEAUX : MEM_TROB

PROCEDURE MAJM

E : tch : tableau intermédiaire contenant un ensemble de valeurs numériques sous la forme alphabétique

trob1 : tableau intermédiaire pouvant contenir un robot

E-S :

S :

BUT : Mettre à jour la partie mémoire où est stocké le robot lors de la modification des caractéristiques d'un bras. L'ancien contenu de la zone est gardé pour le cas où il faudrait le retrouver.

APPEL : OUTIL : DETERMVAL · MEM_COPIEVAL · COPIE-VA_MEM · TRAITTCH

TABLEAUX : MEM_TROB

PROCEDURE MAJS

E : tch : tableau intermédiaire contenant un ensemble de valeurs numériques sous la forme alphabétique

trob1 : tableau intermédiaire pouvant contenir un robot

E-S :

S :

BUT : Mettre à jour la partie mémoire où est stocké le robot lors de la suppression d'un bras. L'ancien contenu de la zone est gardé pour le cas où il faudrait le retrouver.

APPEL : OUTIL : MEM_COPIEVAL_MEM

TABLEAUX : MEM_TROB

PROCEDURE MAJORI

E :

E-S :

S :

BUT : Mettre à jour la zone mémoire contenant les origines des bras lors du changement de la structure du robot

APPEL : TABLEAUX : TORI_MEM · MEM_TORI · MEM_TROB

UNIT OUTILSAI

PROCEDURE TRAIT_EFFAC

E : tdon : un tableau intermédiaire de strings contenant un ensemble de libellés de champs à afficher à l'écran.

i : le numéro du champs traité

E·S : tch : un tableau intermédiaire contenant un ensemble de valeurs en correspondance avec le tableau des libellés

S :

BUT : Effacer le contenu de la donnée i du tableau tch et afficher à l'écran la donnée i des tableaux tdon et tch

APPEL :

PROCEDURE LIRE_DONN

E : ch : le caractère lu

i : le numéro de la donnée traitée

E·S : tch : un tableau de strings contenant un ensemble de valeurs

S :

BUT : Copier le caractère ch à l'endroit adéquat de la ligne i du tableau tch en fonction du caractère lu et du string qui est déjà lu

APPEL :

PROCEDURE TRAIT_ENTER

E : tdon : un tableau intermédiaire de strings contenant un ensemble de libellés de champs à afficher à l'écran.

i : le numéro du champs traité

tch : un tableau intermédiaire contenant un ensemble de valeurs en correspondance avec le tableau des libellés

E·S :

S :

BUT : Traiter la confirmation d'un champ saisi et afficher la donnée saisie, ce qui correspond à la ligne i des tableaux tdon et tch, avant de désigner l'élément suivant comme élément courant

APPEL :

PROCEDURE TRAIT_LIG1

E : tdon : un tableau intermédiaire de strings contenant un ensemble de libellés de champs à afficher à l'écran.

i : le numéro du champs traité

tch1 : un tableau intermédiaire contenant un ensemble de valeurs en correspondance avec le tableau des libellés

E·S :

S :

BUT : Passer à l'élément suivant des tableaux tdon et tch1 et assurer leur affichage à l'écran.

APPEL :

PROCEDURE TRAIT_LIG_1

E : tdon : un tableau intermédiaire de strings contenant un ensemble de libellés de champs à afficher à l'écran.

i : le numéro du champs traité

tch : un tableau intermédiaire contenant un ensemble de valeurs en correspondance avec le tableau des libellés

E-S :

S :

BUT : Passer à l'élément précédent des tableaux tdon et tch et assurer leur affichage à l'écran.

APPEL :

PROCEDURE TRAIT_ENREG

E : tdon : un tableau intermédiaire de strings contenant un ensemble de libellés de champs à afficher à l'écran.

i : le numéro du champs traité

tch1 : un tableau intermédiaire contenant un ensemble de valeurs en correspondance avec le tableau des libellés

lig1, lig2 : messages d'erreur qui sont affichés lorsque la validation detecte une erreur

E-S :

S : numecran : le numéro de l'écran suivant

fin : booleen indiquant si la validation s'est passée correctement

tch : tableau intermédiaire de strings contenant un ensemble de valeurs associées à un bras.

BUT : Valider les données contenue dans le tableau intermédiaire tch1 et mettre à jour le tableau tch. Si la validation s'est déroulé correctement, on passe à l'étape suivante. Sinon, on a l'affichage des messages d'erreur et on revient à la saisie des valeurs.

APPEL :

Unit Dessin.

Déclaration : Procédure DESSINBRAS;

Paramètres

Entrées

NUMBRAS (integer) spécifie le numéro de bras à dessiner.

Prop. : 0 NUMBRAS6

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX0 = 240 (CGA,EGA,VGA)COORDX0 = 270 (HERCULES)

COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY0 = 100 (CGA)COORDY0 = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

Entreé · sortie : Néant

Sortie : Néant

Fonction DESSINBRAS dessine le bras (et l'organe actif qui est considéré comme un bras), portant le numéro NUMBRAS dans la couleur COULEUR, du robot dont le point d'attache avec la surface d'appui est dessiné à l'écran au point de coordonnée x, COORDX0, et de coordonnée y, COORDY0. La longueur du bras dessiné est multipliée par le coefficient UNITE dont la valeur est déterminée en fonction de la carte graphique . Si la valeur de NUMBRAS est telle que l'organe actif est dessiné, DONN indique s'il faut dessiner ou non la trajetoire.

Appel

(DESSIN)

Grosseur_bras,Calcul_pasaug,Calcoorori,Calcoorextact,Affor·gactif, Affliaison,Afftrajectoire,Calcoorext,Affbras1.

(TABLEAUX)

Mem_trob,Mem_tori.

Déclaration : procédure DESSINAXES.

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX0 = 240 (CGA,EGA,VGA)COORDX0 = 270 (HERCULES)COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY0 = 100 (CGA)COORDY0 =160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

Entrée · Sortie : Néant

Sortie : Néant

Fonction DESSINAXES affiche à l'écran le système d'axes (l'axe X,Y et Z) relatif au robot dans la couleur COULEUR. Ce système a comme origine le point dont la coordonnée à l'écran est (COORDX0,COORDY0). La longueur dessinée à l'écran de chaque axe est égale à UNITE.

Appel Néant.

Déclaration : procédure calcoorext

Paramètres

Entrées

INDTBROB (integer) représente l'indice qui parcourt le bras courant (NUMBRAS)du tableau de type TBROB.

VARIATGROSSEUR (integer) indique la grosseur (en nombre de point écran)du bout du bras courant dessiné.

NUMBRAS (integer) spécifie le numéro de bras à dessiner.

Prop. : 0 NUMBRAS6

COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX0 = 240 (CGA,EGA,VGA)COORDX0 = 270 (HERCULES)

COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY0 = 100 (CGA)COORDY0 =160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

Entrée sortie.

Néant.

Sortie

COORDXAF3 (integer) est la coordonnée X du troisième point nécessaire pour dessiner le tapèze représentant le bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF3 (integer) est la coordonnée Y du troisième point nécessaire pour dessiner le tapèze représentant le bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDXAF4 (integer) est la coordonnée X du quatrième point nécessaire pour dessiner le tapèze représentant le bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF4 (integer) est la coordonnée Y du quatrième point nécessaire pour dessiner le tapèze représentant le bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

Fonction CALCOOREXT calcule les coordonnées du troisième et quatrième points nécessaire pour dessiner le trapèze qui représente le bras du robot NUMBRAS.

Appel (TABLEAUX)
Men_trob,Mem_tori

Déclaration : procédure calcoorori

Paramètres

Entrées
INDTBROB (integer) représente l'indice qui parcourt le bras courant (NUMBRAS)du tableau de type TBROB.

VARIATGROSSEUR (integer) indique la grosseur (en nombre de point écran)du bout du bras courant dessiné.

NUMBRAS (integer) spécifie le numéro de bras à dessiner.
Prop. : 0 NUMBRAS6
COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.
Prop :
COORDX0 = 240 (CGA,EGA,VGA)COORDX0 = 270 (HERCULES)

COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.
Prop :
COORDY0 = 100 (CGA)COORDY0 = 160 (EGA,VGA,HERCULES)
UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.
Prop :
UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)
Entrée sortie.
Néant.

Sortie

COORDXAF1 (integer) est la coordonnée X du premier point nécessaire pour dessiner le tapèze représentant le bras du robot ou le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDX0

saire pour dessiner le tapèze représentant le bras du robot ou le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDXAF2 (integer) est la coordonnée X du deuxième point nécessaire pour dessiner le tapèze représentant le bras du robot ou le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF2 (integer) est la coordonnée Y du deuxième point nécessaire pour dessiner le tapèze représentant le bras du robot ou le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

Fonction CALCOORORI calcule les coordonnées du premier et deuxième points nécessaire pour dessiner le trapèze qui représente le bras du robot NUMBRAS ou le triangle représentant l'organe actif.

Appel (TABLEAUX)

Men_trob,Mem_tori

Déclaration : procédure calcoorextact

Paramètres

Entrées

INDTBROB (integer) représente l'indice qui parcourt le bras courant (NUMBRAS)du tableau de type TBROB.

COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX0 =240 (CGA,EGA,VGA)COORDX0 =270 (HERCULES)

COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY0 = 100 (CGA)COORDY0 = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

Entrée sortie.

Néant.
Sortie

COORDXAF3 (integer) est la coordonnée X du troisième point nécessaire pour dessiner le triangle représentant l'organe actif (bras particulier pour l'affichage à l'écran) du robot. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDX0
COORDYAF3 (integer) est la coordonnée Y du troisième point nécessaire pour dessiner le triangle représentant l'organe actif du robot. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDY0

Fonction CALCOOREXTACT calcule les coordonnées du troisième point nécessaire pour dessiner le triangle qui représente l'organe actif du robot.

Appel (TABLEAUX)
Men_trob

Déclaration procedure grosseur_bras

Paramètres

Entrée
COORDZ (real) est la coordonnée Z, dans le référentiel écran du simulateur, du bras courant dessiné.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.
Prop HAUTEUR = 5
DIM (real) est la dimension du bras. Cette dimension est donnée par l'utilisateur lors de la création du robot.Prop DIM = 0.2/0.4/0.6/0.8/1

DIM_MAX (real) est la coordonnée Z maximale en valeur absolue de l'organe actif du robot dans le référentiel écran.
Entrée sortie

Néant
Sortie
GROSSEUR (integer) indique la grosseur du bras.
Prop 0GROSSEUR

Fonction GROSSEUR_BRAS calcule la grosseur de l'origine et de l'extrémité du bras courant en fonction de la coordonneé Z, de l'origine et de l'extrémité, dans le référentiel écran.

Appel Néant.

Déclaration procedure afforgactif

Paramètres
Entrée

COORDXAF1 (integer) est la coordonnée X du premier point nécessaire pour dessiner le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDX0

COORDYAF1 (integer) est la coordonnée Y du premier point nécessaire pour dessiner le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDY0

COORDXAF2 (integer) est la coordonnée X du deuxième point nécessaire pour dessiner le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDX0

COORDYAF2 (integer) est la coordonnée Y du deuxième point nécessaire pour dessiner le triangle représentant l'organe actif. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDY0

COORDXAF3 (integer) est la coordonnée X du troisième point nécessaire pour dessiner le triangle représentant l'organe actif (bras particulier pour l'affichage à l'écran) du robot. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDX0

COORDYAF3 (integer) est la coordonnée Y du troisième point nécessaire pour dessiner le triangle représentant l'organe actif du robot. Cette coordonnée est exprimée en nombre de point écran.
Prop : idem COORDY0

COULEUR (integer) détermine la couleur du bras dessiné.
Prop : COULEUR = 0..15

Entrée · sortie

Néant

Sortie

Néant

Fonction AFFORGACTIF dessine l'organe actif du robot dans la couleur COULEUR. L'organe actif est représenté par un triangle dont les 3 points ont les coordonnées suivantes : COORDXAF1,COORDYAF1

COORDXAF2,COORDYAF2

COORDXAF3,COORDYAF3

Déclaration procedure calcul_dim_max

Paramètres

Entrée

Néant

Entrée sortie

Néant

Sortie

DIM_MAX (real) est la longueur total du robot (bras + organe actif).

Fonction CALCUL_DIM_MAX donne la coordonnée Z maximale en valeur absolue de l'organe actif du robot dans le référentiel écran.

Appel

(TABLEAUX)

Mem_tori,Mem_trob

Déclaration procedure affbras1

Paramètres

Entrée

COORDXAF1 (integer) est la coordonnée X du premier point nécessaire pour dessiner le trapéze représentant le bras courant dessiné.Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF1 (integer) est la coordonnée Y du premier point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDXAF2 (integer) est la coordonnée X du deuxième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF2 (integer) est la coordonnée Y du deuxième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDXAF3 (integer) est la coordonnée X du troisième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF3 (integer) est la coordonnée Y du troisième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDXAF4 (integer) est la coordonnée X du quatrième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDX0

COORDYAF4 (integer) est la coordonnée Y du quatrième point nécessaire pour dessiner le trapèze représentant le bras courant dessiné. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

Entrée · sortie

Néant

Sortie

Néant

Fonction AFFBRAS1 dessine l'organe actif du robot dans la couleur COULEUR. L'organe actif est représenté par un triangle dont les 4 points ont les coordonnées suivantes : COORDXAF1,COORDYAF1

COORDXAF2,COORDYAF2

COORDXAF3,COORDYAF3COORDXAF4,COORDYAF4

Déclaration procedure affliaison

Paramètres

Entrée

COORDX1 (integer) est la coordonnée X du premier point extrémité du bras, point utilisé pour dessiner le trapèze représentant un bras du robot. Cette coordonnée est exprimée en nombre de point écran

du bras, point utilisé pour dessiner le trapèze représentant un bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

COORDX2 (integer) est la coordonnée X du deuxième point extrémité du bras, point utilisé pour dessiner le trapèze représentant un bras du robot. Cette coordonnée est exprimée en nombre de point écran

Prop : idem COORDX0

COORDY2 (integer) est la coordonnée Y du deuxième point extrémité du bras, point utilisé pour dessiner le trapèze représentant un bras du robot. Cette coordonnée est exprimée en nombre de point écran.

Prop : idem COORDY0

NUMBRAS (integer) spécifie le numéro de bras à dessiner.

Prop. : 0 NUMBRAS6

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

GROS1 (integer) est le rayon du cercle dessiné représentant la liaison entre deux bras (ou le dernier bras et l'organe actif) du robot.

Entrée sortie

Néant

Sortie

Néant

Fonction AFFLIAISON dessine un cercle représentant la liaison existant entre deux composants du robot.

Appel

(TABLEAUX)

Mem_trob

Déclaration procedure afftrajectoire

Paramètres

Entrée

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX0 (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX0 = 240 (CGA,EGA,VGA)COORDX0 = 270 (HERCULES)

COORDY0 (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY0 = 100 (CGA)COORDX0160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

Entreé · sortie : Néant

Sortie : Néant

Fontion AFFTRAJECTOIRE dessine la trajectoire suivie par le robot depuis le début de sa séquence de mouvement.

Appel

Néant.

UNIT ECRAN

PROCEDURE ECRAN1

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

fin_dir : booleen indiquant si on termine directement sa session de travail

BUT : Afficher l'écran 1 qui présente le logiciel et commence la partie CREATION d'un robot

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN2

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

robexi : booleen qui indique si on choisit d'utiliser un robot existant ou d'en créer un.

premier : indication servant dans la suite dans le cas où on construit le robot.

BUT : Afficher l'écran2 où on demande à l'utilisateur s'il désire choisir un robot déjà construit ou s'il désire le construire.

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN90

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 90 où l'utilisateur est averti de la perte de toutes les données saisies à cause de l'activation de la touche qu'il vient de faire et où il doit comfirmer ou imfirmer son choix précédent.

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN93

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran93 où l'utilisateur doit indiquer s'il désire que la trajectoire suivie par l'organe actif du robot

qu'il désire charger soit gardée ou non. Cet écran est affiché uniquement lorsqu'on fait appel à un robot existant.
APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN10

E : donn : dataconfig

E·S :

S : par_nom , par_dir : booleens servant à indiquer le choix fait par l'utilisateur

numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 10 où, lorsque l'utilisateur décide de charger un robot existant pour en faire la simulation, il doit indiquer la manière choisie pour sélectionner son robot. Soit il la fait en demandant tous les robots répertoriés pour un utilisateur donné, soit il accède directement au robot en donnant la suite de directory.
APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN101

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

nom : le nom d'utilisateur saisi

gest : booleen indiquant s'il faut faire une gestion du nom saisi

BUT : Afficher l'écran101 où on fait la saisie d'un nom d'utilisateur

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN111

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 111 où on prévient l'utilisateur que le nom d'utilisateur qu'il a donné est inconnu du système et où on demande s'il désire faire une nouvelle saisie.

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN102

E : donn : dataconfig

E·S :

S : numecran : le numero de l'écran suivant

BUT : afficher l'écran 102 où l'utilisateur doit indiquer le chemin complet à suivre pour accéder à un robot particulier.

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN112

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 112 où on prévient l'utilisateur que le chemin d'accès à un robot qu'il a donné est incorrect ou impossible et où on lui demande s'il désire recommencer la saisie d'un chemin

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN3

E : donn : dataconfig

premier : booleen qui indique si on a déjà placer un bras au robot que l'on construit ou si c'est le premier.

E·S :

S : numecran : le numéro de l'écran suivant

maj : caractère indiquant le type de changement que l'on désire apporter à la structure actuelle du robot

BUT : Afficher l'écran3 où l'utilisateur doit choisir le type de changement qu'il désire apporter parmi ceux qui lui sont proposés.

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN5

E : donn : dataconfig

maj : caractère indiquant le type de changement on souhaite apporter à la structure actuelle du robot

E·S :

S : tch tableau intermédiaire où on mémorise dans le premier élément le numéro de bras saisi.

numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 5 où l'utilisateur doit indiquer le numéro du bras qu'il désire traiter . Une gestion de ce numéro est faite en fonction des contraintes liées à la structure actuelle du robot et au type de changement choisi.

APPEL : ECRAHELP : ECRHELP

UNIT ECRANDIF

PROCEDURE ECRAN6

E : coordx, coordy

unite

hauteur

couleur
bornex, borney

Ces données servent lors de la visualisation du robot à l'écran
maj : caractère indiquant le type de changement apporté à la structure du robot

tch : tableau intermédiaire contenant les caractéristiques d'un bras du robot

robexi : booleen indiquant si la visualisation fait suite au chargement d'un robot existant ou à la création d'un nouveau.

donn : dataconfig

E-S : premier : booleen indiquant si on traite le premier bras d'un robot

S : numecran : le numéro de l'écran suivant

BUT : Afficher la structure actuelle d'un robot suite à un changement de structure. La structure précédente est mémorisée pour le cas où on rejette la nouvelle structure. L'utilisateur a la possibilité d'enregistrer ou de rejeter la structure créée du robot. En fonction de son choix, on détermine le numéro de l'écran suivant.

APPEL : DESSIN : DESSINBRAS · DESSINAXES

COORDONN : COORDACTIF · COORDFIXE
OUTIL : MAJA · MAJM · MAJS ·COPIETAB_MEM
ECRAHELP : ECRHELP

LIROBOT : INITDONN

ECRAHELP : ECRHELP

PROCEDURE ECRAN9

E : coordx, coordy

unite

hauteur

couleur

bornex, borney

Ces données servent lors de la visualisation du robot à l'écran

donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Visualiser le robot et permettre de déplacer l'emplacement qu'il occupe à l'écran en ayant recours aux flèches du clavier.

APPEL : DESSIN : DESSINBRAS -DESSINAXES

ECRAHELP : ECRHELP

PROCEDURE ECRAN12

E : nom : le nom de l'utilisateur

donn : dataconfig

E-S :

S : nom_robot : le nom du robot sélectionné

numecran : le numéro de l'écran suivant

BUT : Afficher à l'écran une partie des robots mémorisés sous ce nom d'utilisateur et permettre à l'utilisateur de se déplacer dans la liste des robots en utilisant les flèches. Lorsque le choix est fait, le nom du robot est connu et on passe à l'écran suivant

APPEL : ECRAHELP : ECRHELP

UNIT ECRANSAI

PROCEDURE ECRAN40

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Afficher l'écran 40 où l'utilisateur doit saisir un ensemble de caractéristiques du bras et assurer de leur validité avant de les mémoriser dans tch

APPEL : ECRAHELP : ECRHELP

OUTILSAI : TRAIT_LIG1 · TRAIT_LIG_1 · TRAIT_EFFAC · TRAIT_ENREG · LIRE_DONN

PROCEDURE ECRAN41

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Afficher l'écran 41 dans lequel l'utilisateur définit le type de liaison entre le bras traité et le bras précédent

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN42

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Afficher l'écran 42 dans lequel l'utilisateur définit les variations possibles pour les angles A et B lorsque le type de liaison est une rotule

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN43

E : donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Afficher l'écran 43 dans lequel l'utilisateur définit les variations possibles pour les angles A et B lorsque le type de liaison est une charnière

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN44

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Afficher l'écran 44 dans lequel l'utilisateur définit les axes le long desquels une translation est possible lorsqu'on traite le premier bras.

APPEL : ECRAHELP : ECRHELP

PROCEDURE TRAIT_ORG_ACTIF

E :

E-S :

S : numecran : le numéro de l'écran suivant

tch : tableau intermédiaire contenant un ensemble de strings correspondant à des valeurs numériques.

BUT : Définir l'ensemble des caractéristiques du bras particulier qu'est l'organe actif

APPEL :

UNIT ECRANSUI

PROCEDURE ECRAN7

E : donn : dataconfig

E·S :

S : chang : booleen indiquant si l'on désire changer l'emplacement du robot

numecran : le numéro de l'écran suivant

BUT : Afficher l'écran7 où l'utilisateur peut choisir de déplacer l'emplacement de son robot à l'écran

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN8

E : coordx, coordy

unite
hauteur
couleur

coordx, coordy

Ces données servent lors de la visualisation du robot à l'écran

donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran final où le robot est visualisé dans la structure et position définitive avant de commencer la simulation

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN20

E : donn : dataconfig

E·S :

S : ch : caractère indiquant le choix de l'utilisateur

aide : booleen indiquant si l'utilisateur désire faire appel à un écran d'aide.

BUT : Afficher l'écran 20 où l'utilisateur doit indiquer s'il désire sauver le robot dans un fichier. Sa réponse est placée dans ch

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN21

E : donn : dataconfig

E·S :

S : nomutil : nom de l'utilisateur qui est propriétaire du robot sauvé

nomrob : le nom du robot sauvé

couaoud : cvaractère indiquant la suite qu'il faut donner à cet écran.
c pour sauvetage dans la directory par défaut
a pour abandon du sauvetage
d pour sauvetage dans une autre directory

aide : booleen indiquant que l'utilisateur désire faire appel à un écran d'aide

BUT : Afficher l'écran 21 dans lequel on saisit les renseignements nécessaires pour le sauvetage du robot et déterminer la suite que l'on donne à cet écran

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN22

E : donn : dataconfig

E-S :

S : chemin : string définissant un chemin d'accès à une directory où on sauve le robot

numecran : le numéro de l'écran suivant

BUT : Afficher l'écran22 dans lequel on saisit le chemin de la directory particulière, autre que celle par défaut, où on veut mémoriser le robot

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN14

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran 14 où l'utilisateur est averti que le robot qu'il désire lire n'est pas accessible dans l'état actuel du système

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN13

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran13 où l'utilisateur est averti que le robot qu'il désire mémoriser correspond à l'identification d'un robot déjà existant et où il doit indiquer s'il désire l'écraser ou non

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRAN15

E : donn : dataconfig

E-S :

S : numecran : le numéro de l'écran suivant

BUT : Afficher l'écran15 où l'utilisateur est prevenu que le robot qu'il a créé n'est pas complet puisqu'il ne comprend pas d'organe actif

APPEL : ECRAHELP : ECRHELP

PROCEDURE ECRANFS

E : donn : dataconfig

E-S : fin_com : booleen indiquant si l'on doit continuer à lire des commandes

S : fin_sess : booleen indiquant si l'on désire continuer sa session

BUT : Afficher l'écranfs où l'utilisateur doit indiquer s'il désire continuer sa session ou non

APPEL : ECRAHELP : ECRHELP

UNIT ECRAHELP

PROCEDURE ECRHELP

E : donn : dataconfig

E·S : num_ecr_help : le numéro de l'écran d'aide à afficher

S :

BUT : Gérer le parcours de la structure d'arbre que constitue l'ensemble des écrans d'aide en faisant appel aux différentes procédures d'affichages du contenu des écrans d'aide définis dans l'UNIT GESTHELP

APPEL : GESTHELP : TOUTES LES PROCEDURES

UNIT GESTECRA

PROCEDURE GESTECRA3

E : maj : le type de changement que l'on apporte à la structure du robot

E·S :

S : numecran : le numéro de l'écran correspondant au changement

BUT : Définir le numéro de l'écran en fonction du type de changement apporté à la structure du robot

APPEL :

PROCEDURE GESTECRA101

E : nom : nom d'un utilisateur

donn : dataconfig

E·S :

S : numecran : le numéro de l'écran suivant

BUT :Vérifier l'existance d'un nom d'utilisateur et définir le numéro de l'écran suivant en fonction de l'existance ou nono de l'utilisateur

APPEL : LIROBOT : LECTURE_DIR_COUR

PROCEDURE GESTECRAN102

E : chemin : chemin d'accès à un fichier où est mémorisé un robot

E·S :

S : numécran : le numéro de l'écran suivant

BUT : Verifier si le chemin saisi permet d'accéder à un robot et de le copier dans la zone mémoire réservée à cet effet. En déduire le numéro de l'écran suivant

APPEL : LIROBOT : VERIFICATION_ACCES_FICH_ROBOT · COPIE_ROBOT_DM

PROCEDURE GESTECRAN7

E : maj : le type de changement que l'on désire apporter à la structure du robot

E·S :

S : numecran : le numéro de l'écran suivant

BUT : Vérifier si le robot est complet avant de terminer la partie construction du robot et en déduire le numéro de l'écran suivant.

APPEL : TABLEAUX : MEM_TROB

UNIT GESTHELP

Cette unit assure la gestion des écrans d'aide définis pour le logiciel.

Toutes les procédures ont pour même but d'afficher le contenu d'un fichier qui contient un texte.

Les spécifications de ces procédures étant toutes parfaitement identiques, elles ne seront écrites que pour la première. Les autres procédures seront uniquement citées

L'ensemble des fichiers d'aide constitue une structure d'arbre qu'il est possible de parcourir à partir de tout écran d'aide.

PROCEDURE HFECR1

E : donn : dataconfig

E-S :

S : num_ecr_help : numéro de l'écran d'aide suivant à afficher.

BUT : Afficher le contenu du fichier d'aide numéro 1 et fournir le numéro de l'écran suivant. Ce numéro est fixé en fonction de la touche activée par l'utilisateur à la fin de sa lecture.

APPEL :

PROCEDURE HFECR2

PROCEDURE HFECR21

PROCEDURE HFECR211

PROCEDURE HFECR212

PROCEDURE HFECR213

PROCEDURE HFECR214

PROCEDURE HFECR215

PROCEDURE HFECR216

PROCEDURE HFECR217

PROCEDURE HFECR218

PROCEDURE HFECR22

PROCEDURE HFECR221

PROCEDURE HFECR222

PROCEDURE HFECR223

PROCEDURE HFECR3

PROCEDURE HFECR31

PROCEDURE HFECR32

PROCEDURE HFECR33

PROCEDURE HFECR4

PROCEDURE HFECR41

PROCEDURE HFECR5

PROCEDURE HFECR51

PROCEDURE HFECR52

PROCEDURE HFECR6

PROCEDURE HFECR61

PROCEDURE HFECR62

PROCEDURE HFECR63

Unit Ecransv.

Déclaration procedure ecran1sv

Paramètres

Entrée
DONN (dataconfig) contient les informations relatives à l'environement du simulateur.
Entrée sortie

Néant

Sortie
CH (char) contient la réponse qui s'il faut remplacer un robot existant par le robot courant étant donné que son nom est le même que celui existant déjà.

Prop

CH = O si il faut remplacer le robot existant par le robot courant, sinon CH = N.

Fonction ECRAN1SV signale à l'utilisateur que le nom donné au robot courant est le même qu'un robot existant déjà. Il demande dès lors si le robot courant doit remplacer le robot existant.

Appel Néant

Déclaration procedure ecran2sv

Paramètres

Entrée
DONN (dataconfig) contient les informations relatives à l'environement du simulateur.
Entrée sortie

Néant
Sortie

Néant

Fonction ECRAN2SV signale que le sous-directory choisi par l'utilisateur pour sauver le robot est un sous-directory réservé pour le simulateur.

Appel Néant

Déclaration procedure ecran3sv

Paramètres

Entrée
DONN (dataconfig) contient les informations relatives à l'environement du simulateur.

Entrée sortie

Néant

Sortie

Néant

Fonction ECRAN3SV signale que le chemin d'accès choisi par l'utilisateur au bout duquel le robot sera sauvé est syntaxiquement incorrect..

Appel Néant

Unit Initiaec.

Déclaration : Procédures traitcga,traitega,traitvga,traitherc;

Paramètres

Entrées

Néant

Entreé · sortie
Néant

Sortie

COORDX (integer) est la coordonnée x du plot (point·écran) repré·sentant le point d'attache entre le robot et la surface d'appui. CO·ORDX est exprimé en nombre de point écran.

Prop
COORDX = 240 (CGA,EGA,VGA)
COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point·écran) repré·sentant le point d'attache entre le robot et la surface d'appui. COOR·DY est exprimé en nombre de point écran.

Prop
COORDY = 100 (CGA)
COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affi·chage d'un bras. Sa valeur est fonction de la carte graphique.

Prop
UNITE = 50 (CGA)
UNITE = 100 (EGA,VGA,HERCULES)

HAUTEUR (integer) donne la hauteur du trapèze, dessiné à l'écran, représentant un bras dont l'angle A vaut 90 degrès (l'angle B n'in·fluence pas la hauteur du trapèze)

Prop
HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BOR·NEX est exprimé en nombre de point écran.

Prop
BORNEX = 480 (CGA,EGA,VGA)
BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop
BORNEY = 200 (CGA)
BORNEY = 350 (EGA,VGA)
BORNEY = 320 (HERCULES)

Fonction Les paramètres en sortie sont assez explicites pour comprendre ce que fait les différentes fonctions.

Appel Néant

Déclaration procedure initecran

Paramètres

Entrée
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entrée sortie

Néant

Sortie

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. COORDX est exprimé en nombre de point écran.

Prop
COORDX = 240 (CGA,EGA,VGA)
COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. COORDY est exprimé en nombre de point écran.

Prop
COORDY = 100 (CGA)
COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop
UNITE = 50 (CGA)
UNITE = 100 (EGA,VGA,HERCULES)

HAUTEUR (integer) donne la hauteur du trapèze, dessiné à l'écran, représentant un bras dont l'angle A vaut 90 degrès (l'angle B n'influence pas la hauteur du trapèze)

Prop
HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop
BORNEX = 480 (CGA,EGA,VGA)
BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop
BORNEY = 200 (CGA)
BORNEY = 350 (EGA,VGA)
BORNEY = 320 (HERCULES)

WINDX (integer) donne la coordonnée X du coin supérieur gauche de la fenêtre réservée au programme de l'utilisateur.

Fonction INITECRAN initialise le mode graphique et détermine et en fonction de la carte graphique reconnue, détermine le fenêtrage (BORNEX,BORNEY), les paramètres utiles pour dessiner le robot (UNITE,HAUTEUR,COORDX, COORDY).

Appel

(INITIAEC)
Traitcga,Traitega,Traitvga,Traitherc
(GRAPH)
Initgraph, Detect

Déclaration procedure copie_fichier_francais

Paramètre

Entrée
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entrée sortie

Néant

Sortie

Néant

Fonction COPIE_FICHIER_FRANCAIS charge les fichiers textes français des différents écrans présents sur floppy disk ou sur le hard disk vers le virtual disk.

Appel
Néant.

Déclaration procedure initlangue

Paramètres

Entrée
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entrée sortie
Néant

Sortie

Néant
Fonction INITLANGUE détermine la langue des fichiers textes des différents écrans qui seront utilisés.

Appel

(INITIAEC)
Copie_fichier_français

Déclaration procedure inittraj

Paramètres

Entrée
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entrée sortie

Néant

Sortie
Néant

Fonction INITTRAJ initialise le fichier qui mémorisera les différents points de la trajectoire suivie par l'organe actif du robot.

Appel Néant.

Déclaration procedure initdirectory

Paramètres

Entrée
Néant

Entrée sortie

Néant
Sortie
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Fonction INITDIRECTORY complète l'environement du simulateur en mémorisant le directory dans lequel il se trouve.

Appel Néant.

Unit Initsim.

Déclaration : procedure INITSIMR

Paramètres
Entrée

Néant

Entreé · sortie

DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Prop
DONN contient en plus le directory dans lequel se trouve le code exécutable du simulateur.

Sortie

COORDX (integer) est la coordonnée x du plot (point·écran) représentant le point d'attache entre le robot et la surface d'appui. COORDX est exprimé en nombre de point écran.

Prop
COORDX = 240 (CGA,EGA,VGA)
COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point·écran) représentant le point d'attache entre le robot et la surface d'appui. COORDY est exprimé en nombre de point écran.

Prop
COORDY = 100 (CGA)
COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop
UNITE = 50 (CGA)
UNITE = 100 (EGA,VGA,HERCULES)

HAUTEUR (integer) donne la hauteur du trapèze, dessiné à l'écran, représentant un bras dont l'angle A vaut 90 degrès (l'angle B n'influence pas la hauteur du trapèze)

Prop
HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop
BORNEX = 480 (CGA,EGA,VGA)

BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop

BORNEY = 200 (CGA)

BORNEY = 350 (EGA,VGA)

BORNEY = 320 (HERCULES)

COULEUR (integer) est la couleur dans lequel le robot sera dessiné à l'écran.

WINDX (integer) donne la coordonnée X du coin supérieur gauche de la fenêtre réservée au programme de l'utilisateur.

Fonction INITSIMR s'occupe d'assurer les conditions nécessaires à une exécution du simulateur. Ces conditions sont les suivantes :

- le mode graphique adéquat initialisé
- le chargement dans le disque virtuel des fichiers textes relatifs au écran.
- l'initialisation du fichier contenant les coordonnées des points de la trajectoire que suivra l'organe actif du robot. Cette initialisation se fait si l'information lue dans le fichier de configuration indique que la trajectoire que suivra l'organe actif du robot sera mémorisée.
- mémorisation du directory dans lequel se trouve le code exécutable du simulateur.
- initialisation de la couleur dans laquelle le robot sera dessiné.

Appel

(INITIAEC)

Initecran, Initdirectory, Initlangue, Initcouleur, Inittraj.

UNIT LECTSIMR

PROCEDURE LECTURE

E : coordx,coordy

unite

hauteur

couleur

bornex, borney

Ces données servent lors de la visualisation du robot à l'écran

donn : dataconfig

E·S :

S :

BUT : Assurer la gestion de la partie création du robot dont on désire simuler le comportement

APPEL : ECRAN : ECRAN1 · ECRAN2 · ECRAN90 · ECRAN91 · ECRAN93 · ECRAN3 · ECRAN5 · ECRAN10 · ECRAN101 · ECRAN102 · ECRAN111 · ECRAN112

ECRANDIF : ECRAN6 · ECRAN9 · ECRAN12

ECRANSAI : ECRAN40 · ECRAN41 · ECRAN42 · ECRAN43 · ECRAN44 · TRAIT_ORG_ACTIF

ECRANSUI : ECRAN7 · ECRAN8 · ECRAN15

LIROBOT : INITDONN · LECTURE_ROBOT

LIREDONN : LIREAXE · LIRANGL · LIREPOS

UNIT LIROBOT

PROCEDURE LIRE_FICH_UTILIS

E : nom_utilis : nom d'un utilisateur du logiciel

nom_robot : nom d'un robot

E-S :

S : acces_fich_robot : nom complet du chemin où est placé le robot dont le nom est donné et dont l'utilisateur propriétaire est connu également

numecran : numéro de l'écran suivant. Ce numéro varie selon le fait que l'on a trouvé ou non le chemin cherché

BUT : Trouver le nom de la directory où est mémorisé le robot dont le nom et le nom de l'utilisateur est donné.

APPEL :

PROCEDURE COPIE_ROBOT_DM

E : chemin_robot : chemin complet pour accéder au fichier contenant le robot désiré.

E-S :

S : numecran : numéro de l'écran suivant

BUT : Charger le robot à partir du fichier dont le chemin d'accès est donné

APPEL : TABLEAUX : TROB_MEM

PROCEDURE LECTURE_DIR_COUR

E : donn : dataconfig

E-S :

S : dir_cour : nom de la directory par défaut définie lors de la configuration du logiciel

BUT : Determiner le nom de la directory par défaut

APPEL :

PROCEDURE VERIFICATION_ACCES_FICH_ROBOT

E : acces_fich_robot : chemin complet pour accéder à un robot

E-S :

S : acess_ok : booleen indiquant si l'accès est possible

BUT : Vérifier si le fichier dont le chemin d'accès est donné est réellement accssible

APPEL :

PROCEDURE LECTURE_ROBOT

E : nom_utilis : nom d'un utilisateur

nom_robot : nom d'un robot

traj_oui : booleen indiquant si le fichier de la trajectoire suivie lors de la dernière simulation doit être chargé en même temps

donn : dataconfig

E·S :

S : numecran : le numéro de l'écran qui suit cette procédure lors de la phase de création du robot que l'on désire utiliser

BUT : Charger le fichier contenant le robot de nom donné et dont l'utilisateur est connu également. Charger si nécessaire le fichier de la dernière trajectoire dans le disque virtuel.

APPEL : LIROBOT : LECTURE_DIR_COUR · LEC· TURE_FICH_UTILIS · VERIFICATION_ACCES_ROBOT

Unit Sauvrob.

Déclaration : Procédure sauvetage_robot

Paramètres
Entrées

NOMUTILIS (str8) est le nom de l'utilisateur.

NOMROBOT (str8) est le nom du robot.

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop
Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

AUTRE (boolean) est un indicateur. Il détermine si le sauvetage se fait dans le directory par défaut.

DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entreé · sortie

SAUVETAGE_REUSSI (boolean) indique si le sauvetage s'est déroulé sans problèmes.

Prop
Sauvetage réussi = SAUVETAGE_REUSSI = true
Sauvetage manqué car problème(s) = SAUVETAGE_ REUSSI = false

Sortie

Néant.

Fonction SAUVETAGE_ROBOT sauve le robot NOMROBOT de l'utilisateur NOMUTILIS dans le sous-directory CHEMINROBOT. Si le sauvetage est réussi, SAUVETAGE_REUSSI est vrai. SAUVETAGE_ROBOT sauve également le fichier des coordonnées de la trajectoire suivie par l'organe actif du robot. Ce fichier est également sauvé dans le sous-directory CHEMINROBOT.

Appel
(SAUVROB)

Lecture_chemin,Verification_chemin,Robot_existe,Creation_chemin,Ss_directory_ok,Ajout_nom_utilis,Creation_fich_robot,

Copie_robot_dm

(ECRANSV)

Ecransv2,Ecransv3

Déclaration procedure verification_chemin

Paramètres

Entrée

DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

CHEMINROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop

Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

Entrée sortie

NUMECRANBIS (integer) désigne l'étape suivante à réaliser pour sauver le robot lorsque la procédure se termine.

Prop

NUMECRANBIS = 0 si un problème survient lors du passage dans VERIFICATION_CHEMIN.

Sinon, NUMECRANBIS = NUMECRANBIS + 1

Sortie

TDIR (tabdir) est un tableau qui contient toutes les parties du chemin d'accès à l'endroit où sera sauvé le robot. Si tous ces parties étaient assemblées, elles formeraient un chemin d'accès syntaxiquement et sémantiquement correcte.

INDTDIR (integer) est l'index de TDIR

Fonction VERIFICATION_CHEMIN teste syntaxiquement et sémantiquement le chemin d'accès (CHEMINROBOT) à l'endroit où sera sauvé le robot. Si le test s'avère correct, TDIR contient les différents sous-directory qui forment le chemin d'accès; et dans ce cas, l'étape suivante du sauvetage du robot sera exécutée (NUMBRAS + 1).

Appel

(SAUVROB)

Lecture_chemin_interdit.

Déclaration procedure creation_chemin

Paramètres

Entrée
TDIR (tabdir) est un tableau qui contient toutes les parties du chemin d'accès à l'endroit où sera sauvé le robot. Si tous ces parties étaient assemblées, elles formeraient un chemin d'accès syntaxiquement et sémantiquement correcte.

INDTDIR (integer) est l'index de TDIR
Entrée sortie
Néant

Sortie

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop
Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

Fonction CREATION_CHEMIN crée le chemin d'accés à partir du contenu de TDIR. Ce chemin étant créé, il est mémorisé dans CHEMINROBOT.

Appel

Néant.

Déclaration procedure lecture_directory_cour

Paramètres
Entrée

DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

Entrée sortie

Néant

Sortie

DIR_COUR (str74) contient le directory où se trouve le code exécutable du simulateur.

Fonction LECTURE_DIRECTORY_COUR mémorise dans DIR_COUR le directory où se trouve le code exécutable du simulateur.
Appel Néant.

Déclaration procedure ajout_nom_utilis

Paramètres

Entrée

NOMUTILIS (str8) est le nom de l'utilisateur.

NOMROBOT (str8) est le nom du robot.

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop

Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

DIR_COUR (str74) contient le directory où se trouve le code exécutable du simulateur.

Entrée sortie

Néant

Sortie

Néant

Fonction AJOUT_NOM_UTILIS ajoute l'utilisateur NOMUTILIS, s'il n'est pas connu du système, au fichier des utilisateurs du simulateur présent dans le sous directory "utilis" du directory DIR_COUR (directory où se trouve le code exécutable du simulateur). Pour cet utilisateur existant ou nouvellement créé, il lui associe le robot NOMROBOT ainsi que le chemin d'accès CHEMINROBOT pour y accéder

Appel Néant.

Déclaration procedure delete_file

Paramètres

Entrée

FICHIER_EFFACE (str74) contient le chemin d'accès ainsi que le nom du fichier qu'il faut effacer.

Entrée sortie

Néant

Sortie
STATUS (byte) indique si l'effacement du fichier s'est réalisée sans problèmes.

Prop
Si pas de problèmes : STATUS = 0
Sinon STATUT = contenu du registre AX.

Fonction DELETE_FILE efface le fichier dont le chemin d'accès et son nom son contenu dans FICHIER_EFFACE. Si une erreur quelconque se déroule pendant l'effacement, cela est signalé par STATUS.

Appel Néant

Déclaration procedure effacement_robot

Paramètres
Entrée

NOMROBOT (str8) est le nom du robot.

ACCES_UTILIS (str 74) contient le chemin d'accès au robot NOMROBOT qui va être effacer.

Entrée sortie

Néant

Sortie

Néant

Fonction EFFACEMENT_ROBOT efface le robot NOMROBOT se trouvant au sous-directory ACCES_UTILIS.

Appel

(SAUVROB)

Delete_file.

Déclaration procedure robot_existe

Paramètres

Entrée
DONN (dataconfig) contient toutes les informations relatives à l'environement du simulateur.

NOMUTILIS (str8) est le nom de l'utilisateur.

NOMROBOT (str8) est le nom du robot.

DIR_COUR (str74) contient le directory où se trouve le code exécutable du simulateur.

Entrée sortie

NUMECRANBIS (integer) désigne l'étape suivante à réaliser pour sauver le robot lorsque la procédure se termine.

Prop

NUMECRANBIS = 0 si un problème survient lors du passage dans ROBOT_EXISTE.

Sinon, NUMECRANBIS = NUMECRANBIS + 1

Sortie

Néant

Fonction ROBOT_EXISTE vérifie si le robot NOMROBOT de l'utilisateur NOMUTILIS existe dans le fichier contenant tous les utilisateurs connus du simulateur.

Appel

(SAUVROB)

Effacement_robot.

Déclaration procedure ss_directory_ok

Paramètres

Entrée

NOMUTILIS (str8) est le nom de l'utilisateur.

RETOUR_PROG (str74) contient le directory où se trouve le code exécutable du simulateur.

Entrée sortie

NUMECRANBIS (integer) désigne l'étape suivante à réaliser pour sauver le robot lorsque la procédure se termine.

Prop

NUMECRANBIS = 0 si un problème survient lors du passage dans SS_DIRECTORY_OK.

Sinon, NUMECRANBIS = NUMECRANBIS + 1

Sortie

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop

Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

Fonction SS_DIRECTORY_OK accède au ss-directory où les data relatives au robot seront sauvées, si ce ss-directory existe, sinon le crée. CHEMINROBOT contient l'entierté du chemin d'accès, syntaxiquement et sémantiquement correct, aux data relatives au robot qui sera sauvé.

Appel Néant.

Déclaration procedure creation_fich_robot

Paramètres

Entrée
NOMUTILIS (str8) est le nom de l'utilisateur.

NOMROBOT (str8) est le nom du robot.

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop
Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

Entrée sortie

NUMECRANBIS (integer) désigne l'étape suivante à réaliser pour sauver le robot lorsque la procédure se termine.

Prop
NUMECRANBIS = 0 si un problème survient lors du passage dans CREATION_FICH_ROBOT.
Sinon, NUMECRANBIS = NUMECRANBIS + 1.

Sortie

Néant

Fonction CREATION_FICH_ROBOT crée le fichier NOMROBOT dans le sous-directory CHEMINROBOT.

Appel Néant.

Déclaration procedure hometrob

Paramètres

Entrée
Néant

Entrée sortie

Néant

Sortie

Néant

Fonction HOMETROB remplace les valeurs courantes et intermédiaires des angles A et B par leur valeur initiale. Il change les valeurs des coordonnées X,Y,Z (dans le référentiel actif du robot et dans le référentiel écran) du point d'appui du robot sur la surface de sorte qu'il coïncide avec l'origine de ces deux référentiels.

Appel

(TABLEAUX)

Trob_mem, Tori_mem

Déclaration procedure copie_robot_md

Paramètres

Entrée

CHEMINTROBOT (str74) est le chemin au bout duquel les data relatives au robot seront sauvées, si le chemin est syntaxiquement et sémantiquement correcte.

Prop

Le dernier sous-directory de CHEMINROBOT ne peut être équivalent à UTILIS ou TXT.

Entrée sortie

NUMECRANBIS (integer) désigne l'étape suivante à réaliser pour sauver le robot lorsque la procédure se termine.

Prop

NUMECRANBIS = 0 si un problème survient lors du passage dans COPIE_ROBOT_MD.

Sinon, NUMECRANBIS = NUMECRANBIS + 1

Sortie

Néant

Fonction COPIE_ROBOT_DM copie les data relatives au robot dans le sous-directory CHEMINROBOT.

Appel

(SAUVROB)

Hometrob

(TABLEAUX)

Mem_trob,Mem_tori

Unit Sauvetag

Déclaration procedure sauver

Paramètres

Entrée
DONN (dataconfig) contient les informations sur l'environement du simulateur.

Entrée sortie

Néant

Sortie

Néant

Fonction L'utilisateur peut ne pas vouloir sauver son robot, mais dans le cas contraire, il sera amené à répondre à une série de questions que le simulateur doit connaître pour sauver correctement le robot courant. SAUVER guide l'utilisateur dans la saisie de ces informations.
Appel

(SAUVROB)
Sauvetage_robot
(ECRANSUI)
Ecran20, Ecran21, Ecran22.

Unit Lirecmd.

Déclaration procedure lire_commande

Paramètres

Entrée

Néant
Entrée sortie

Néant
Sortie
COMMANDE (cmde) contient toutes les informations nécessaires à l'exécution du mouvement désiré.

Fonction LIRECMD lit les informations relatives au mouvement que le simulateur doit exécuter et les sauve dans COMMANDE.

Appel Néant.

Unit Interpre

Déclaration procedure interpreteur_commande

Paramètres
Entrée

COMMANDE (cmde) contient toutes les informations nécessaires à l'exécution du mouvement désiré.

DONN (dataconfig) contient toutes les informations relatives à l'environnement du simulateur.

Entrée sortie

Néant

Sortie

CORRECT (boolean) indique si les informations formant la commande sont sémantiquement correctes. (Pour savoir ce que représente la sémantique de chaque commande, se référer au chapitre concernant la communication entre deux programmes)

Prop
CORRECT = true si les informations formant la commande sont sémantiquement correctes, sinon CORRECT = false.

MOUV (char) représente la nature du mouvement à exécuter.

Prop
Cfr le chapitre "Communication entre deux programmes"

ANGLE (char) repésente l'angle ou l'axe (X,Y,Z) concerné par le mouvement.

Prop.

Cfr le chapitre "Communication entre deux programmes"

SIGNE (char) repésente le "sens" du mouvement.

Prop.

SIGNE = "+" ou "-".

Exemple

Une rotation de + 45 degrés a un mouvement opposé à une rotation de - 45 degrés.

NUMBRAS (longint) est le numéro du bras du robot qui réalise le mouvement demandé.

Prop.

Cfr le chapitre "Communication entre deux programmes"

PAS (longint) est l'ampleur du mouvement.

Prop.

Cfr le chapitre "Communication entre deux programmes"

Fonction : INTERPRETEUR_COMMANDE teste la sémantique de la commande contenue dans COMMANDE. S'il se vérifie, CORRECT = true et MOUV contient la nature du mouvement à exécute, ANGLE représente l'angle ou l'axe selon lequel le mouvement sera réalisé, SIGNE indique le sens du mouvement, NUMBRAS est le numéro du bras qui réalise le mouvement et PAS indique l'ampleur du mouvement; sinon CORRECT = false et le contenu des 5 autres paramètres en sortie est indéterminé.

Appel : Néant

PROCEDURE ECRANTRA

E : couleur
coordx, coordy

hauteur

unite

bornex, borney

Ces données servent lors de la visualisation d'un robot à l'écran

donn : dataconfig

E·S :
S :
BUT : Afficher une à une à l'écran les coordonnées,selon un format spécifique, de tous les points par lesquels est passé l'organe actif lors de la simulation. Cette procedure permet également de passer d'un point à l'autre.

APPEL : DESSIN : DESSINBRAS · DESSINAXES

PROCEDURE IMPRIMTRA

E : donn : dataconfig

E·S :

S :

BUT : Imprimer sur papier selon un format spécifique les coordonnées de tous les points par lesquels est passé l'organe actif au cours de la simulation.

APPEL :

Unit Anglxyz.

Déclaration : Procédure ROTAT1_0;

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX = 240 (CGA,EGA,VGA)COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY = 100 (CGA)COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.

Prop HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop

BORNEX = 480 (CGA,EGA,VGA)

BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop

BORNEY = 200 (CGA)

BORNEY = 350 (EGA,VGA)

BORNEY = 320 (HERCULES)

Entreé · sortie : Néant

Sortie : Néant
Fonction ROTAT1_0 change le point de vue à l'aide des touches prévues à cet effet.

Flèche haut · bas : changement de point de vueselon l'axe X
Flèche droite · gauche : changement de point de vue selon l'axe Y
Pgup · pgdn : changement de point de vue selon l'axe Z
Remarque : Tous les paramètres en entrée, execpté DONN, sont utilisés pour dessiner le mouvement réalisé par le robot. Ils ne sont pas utilisés en tant que tels dans la procédure mais ils sont utilisés par les procédures que ROTAT1_0 appelle.

Appel

(VARANGLE)
Variat1
(DESSIN)
Dessinaxes, Dessinbras
(COORDONN)
Coordori, Coordaxe,Coordfixe,Majtrajaxe.

Unit X_Y_Z.

Déclaration: Procédure ROTAT1_1;

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop: COULEUR = 0..15

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX = 240 (CGA,EGA,VGA) COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY = 100 (CGA) COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA, VGA, HERCULES) UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.

Prop : HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre (une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop

BORNEX = 480 (CGA,EGA,VGA)

BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop:

BORNEY = 200(CGA)
BORNEY = 350(EGA, VGA)
BORNEY = 320(HERCULES)

PAS (longint) est l'ampleur du changement de point de vue. Elle exprimée en nombre de degré.

Entreé-sortie: Néant

Sortie: Néant

Fonction ROTAT1_1 change le point de vue de PAS degré selon l'axe lu dans dans la zone mémoire réservée pour le passage de commande.

Remarque. Ce changement de point de vue ne se fait plus à l'aide de touches du clavier mais via une commande envoyée par le programme de l'utilisateur et mémorisée dans la zone mémoire prévue à cet effet. Tous les paramètres en entrée, execpté DONN, sont utilisés pour dessiner le mouvement réalisé par le robot. Ils ne sont pas utilisés en tant que tels dans la procédure mais ils sont utilisés par les procédures que ROTAT1_0 appelle.

Appel

(DESSIN)

Dessinaxes, Dessinbras

(COORDONN)

Coordori, Coordaxe, Coordfixe, Majtrajaxe.

Unit Rotation

Déclaration : Procédure ROTAT2;

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX = 240 (CGA,EGA,VGA)COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY = 100 (CGA)COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.

Prop HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop

BORNEX = 480 (CGA,EGA,VGA)

BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop

BORNEY = 200 (CGA)

BORNEY = 350 (EGA,VGA)
BORNEY = 320 (HERCULES)

PAS (longint) est l'ampleur de la rotation. Elle exprimée en nombre de degré.

ANGLE (char) repésente l'angle de la rotation.
Prop
Cfr le chapitre "Communication entre deux programmes"

SIG (char) repésente le "sens" du mouvement.
Prop
SIGNE = "+" ou "-".

Exemple
Une rotation de + 45 degrés a un mouvement opposé à une rotation de - 45 degrés.

NUMBRAS (longint) est le numéro du bras du robot qui réalise la rotation demandée.
Prop
Cfr le chapitre "Communication entre deux programmes"

Entreé - sortie : Néant

Sortie : Néant

Fonction ROTAT2 exécute une rotation d'un angle ANGLE du bras NUMBRAS du robot. L'ampleur de cette rotation vaut PAS et son sens est déterminé par SIG.

Remarque. COORDX,COORDY,UNITE, HAUTEUR, COULEUR, BORNEX, BORNEY sont utilisés pour dessiner le mouvement réalisé par le robot. Ils ne sont pas utilisés en tant que tels dans la procédure mais ils sont utilisés par les procédures que ROTAT2 appelle.

Appel

(DESSIN)
Dessinaxes, Dessinbras

(COORDONN)
Coordori, Coordaxe,Coordfixe,Majtraj.

Unit Rotbras;

Déclaration : Procédure ROTAT4;

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop : COULEUR = 0..15

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX = 240 (CGA,EGA,VGA)COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY = 100 (CGA)COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100 (EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.

Prop HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre(une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop

BORNEX = 480 (CGA,EGA,VGA)

BORNEX = 540 (HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop

BORNEY = 200 (CGA)

BORNEY = 350 (EGA,VGA)
BORNEY = 320 (HERCULES)

PAS (longint) est l'ampleur du pivotabe. Elle exprimée en nombre de degré.

ANGLE (char) repésente l'angle du pivotage.

Prop

Cfr le chapitre "Communication entre deux programmes"

SIG (char) repésente le "sens" du mouvement.

Prop

SIGNE = "+" ou "-".

Exemple

Un pivotage de + 45 degrés a un mouvement opposé à un pivotage de - 45 degrés.

NUMBRAS (longint) est le numéro du bras du robot qui réalise le pivotage demandé.

Prop

Cfr le chapitre "Communication entre deux programmes"

Entreé - sortie : Néant

Sortie : Néant

Fonction ROTAT4 exécute un pivotage d'un angle ANGLE du bras NUMBRAS du robot. L'ampleur de ce pivotage vaut PAS et son sens est déterminé par SIG.

Remarque. COORDX,COORDY,UNITE, HAUTEUR, COULEUR, BORNEX, BORNEY sont utilisés pour dessiner le mouvement réalisé par le robot. Ils ne sont pas utilisés en tant que tels dans la procédure mais ils sont utilisés par les procédures que ROTAT4 appelle.

Appel

(DESSIN)

Dessinaxes, Dessinbras

(COORDONN)

Coordori, Coordaxe,Coordfixe,Majtraj.

Unit Translat

Déclaration: Procédure ROTAT3

Paramètres

Entrées

COULEUR (integer) détermine la couleur du bras dessiné.

Prop: COULEUR = 0..15

COORDX (integer) est la coordonnée x du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDX = 240 (CGA,EGA,VGA)COORDX = 270 (HERCULES)

COORDY (integer) est la coordonnée y du plot (point-écran) représentant le point d'attache entre le robot et la surface d'appui. Sa valeur est fonction de la carte graphique.

Prop :

COORDY = 100 (CGA)COORDY = 160 (EGA,VGA,HERCULES)

UNITE (integer) est un coefficient qui détermine la longueur d'affichage d'un bras. Sa valeur est fonction de la carte graphique.

Prop :

UNITE = 100(EGA,VGA,HERCULES)UNITE = 50 (CGA)

DONN (dataconfig) contient les informations qui permettent de savoir s'il faut dessiner ou non la trajectoire.

HAUTEUR (integer) représente la hauteur, la grosseur (en plot de l'écran) d'un bras dans une position telle que l'angle A vaut 90 quelque soit la valeur de l'angle B.

Prop HAUTEUR = 5

BORNEX (integer) donne la coordonnée X du coin supérieur droit de la fenêtre (une partie de l'espace écran) réservée au simulateur. BORNEX est exprimé en nombre de point écran.

Prop

BORNEX = 480(CGA,EGA,VGA)

BORNEX = 540(HERCULES)

BORNEY (integer) donne la coordonnée Y du coin inférieur droit de la fenêtre réservée au simulateur. BORNEY est exprimé en nombre de point écran.

Prop

BORNEY = 200 (CGA)

BORNEY = 350 (EGA, VGA)

BORNEY = 320 (HERCULES)

PAS (longint) est l'ampleur de la translation. Elle exprimée en nombre de point écran (plot au sens du langage PASCAL).

AXE (char) l'axe (X, Y, Z) suivant lequel la translation est effectuée.

Prop.

Cfr le chapitre "Communication entre deux programmes"

SIG (char) repésente le "sens" du mouvement.

Prop.

SIGNE = "+" ou "-".

Une translation de + 45 a un mouvement opposé à une translation de - 45.

Entreé - sortie: Néant

Sortie: Néant

Fonction ROTAT3 exécute un translation du robot selon l'axe AXE. L'ampleur de cette translation vaut PAS et son sens est déterminé par SIG.

Remarque. COORDX, COORDY, UNITE, HAUTEUR, COULEUR, BORNEX, BORNEY sont utilisés pour dessiner le mouvement réalisé par le robot. Ils ne sont pas utilisés en tant que tels dans la procédure mais ils sont utilisés par les procédures que ROTAT3 appelle..

Appel

(DESSIN)

Dessinaxes, Dessinbras

(COORDONN)

Coordori, Coordaxe, Coordfixe, Majtraj.

Bibliographie

1. "Un environnement d'aide à la programmation d'un robot : la couche objet", mémoire présenté par Philippe Berthet, 1986

2. "Current trends in graphics and animation", Brian Wyvill, Department of Computer Science, University of Calgary. Article édité dans : "Artificial intelligence, Graphics and Simulation", edited by Graham Birtwistle.

3. "Simulating a robot using graphics and animation", James R. Parker, Department of Computer Science, The university of Calgary. Article édité dans : "Artificial intelligence, Graphics and Simulation", edited by Graham Birtwistle.

4. "An assembly modelling system for dynamic and kinematic analysis", S H Kim and K Lee. Article édité dans CAD (Computer-Aided-Designed), Butterworths, Volume 21, numero 1, Janvier - Février 1989.

5. Chapitre 13 édité du cours d'intelligence artificielle dispensé par Monsieur J. Barreto au Portugal en 1988, "Qualitative simulation".

6. "Dos, Power User's guide", Kris Jamsa, Osborne McGraw - Hill, Berkeley, California, 1988.

7." Turbo Pascal : Advanced Programmer's Guide", Stephen K.O'Brien, Borland. Osborne/McGraw Hill, 1988

8. "Fundamentals of Robotics, Theory and Applications", Larry Heath, Indiana State University, 1985.

9. "Graphisme 3D sur votre micro - ordinateur", Jean-Louis Vuldy

10. "Robots, Principes et contrôle", Claude-Yves Vilbet, 1987. Edition Ellipses.

11. "Robotique générale", Alain Pruski, 1988, Edition Ellipses.

12. "State of the art and predictions for AI and robotics", in "Robotics and AI", Nato ASI series, 1984

13. "Turbo Pascal Version 5", Borland, 1989.

14. "Turbo C, Version 2", Borland